5-2013

Unsupervised and semi-supervised fuzzy clustering with multiple kernels.

Naouel Baili
University of Louisville

UNSUPERVISED AND SEMI-SUPERVISED FUZZY CLUSTERING WITH MULTIPLE KERNELS

By

Naouel Baili
B.S. C.S., Polytechnic School of Tunisia, 2005

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2013

UNSUPERVISED AND SEMI-SUPERVISED FUZZY CLUSTERING WITH MULTIPLE KERNELS

By

Naouel Baili
B.S. C.S., Polytechnic School of Tunisia, 2005

A Dissertation Approved

on April 22, 2013

by the following Dissertation Committee:

Dr. Hichem Frigui - Dissertation Director

Dr. Ryan S. Gill

Dr. Olfa Nasraoui

Dr. Eric C. Rouchka

Dr. Roman V. Yampolskiy

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation and respect to my advisor Dr. Hichem Frigui for his guidance, support and patience over the course of this work. I would also like to thank Dr. Olfa Nasraoui for her constant interest and encouragement towards my work.

I would like to express my deepest gratitude to my parents for their love and moral support through the duration of my studies. I am also heartily grateful to my husband for his understanding and support.

Finally, I would like to dedicate this dissertation to my son Yassin.

ABSTRACT

UNSUPERVISED AND SEMI-SUPERVISED FUZZY CLUSTERING WITH MULTIPLE KERNELS

NAOUEL BAILI

APRIL 22, 2013

For real-world clustering tasks, the input data is typically not easily separable due to the highly complex data structure or when clusters vary in size, density and shape. Recently, kernel-based clustering has been proposed to perform clustering in a higher-dimensional feature space spanned by embedding maps and corresponding kernel functions. Although good results were obtained using the Gaussian kernel function, its performance depends on the selection of the scaling parameter among an extensive range of possibilities. This step is often heavily influenced by prior knowledge about the data and by the patterns we expect to discover. Unfortunately, it is often unclear which kernels are more suitable for a particular task. The problem is aggravated for many real-world clustering applications, in which the distributions of the different clusters in the feature space exhibit large variations. Thus, in the absence of a priori knowledge, a single kernel selected from a predefined group is sometimes insufficient to represent the data.

One way to learn optimal scaling parameters is through an exhaustive search of one optimal scaling parameter for each cluster. However, this approach is not practical since it is computationally expensive, especially when the data includes a large number of clusters and when the dynamic range of possible values of the scaling parameters is large. Moreover, the evaluation of the resulting partition in order to select the optimal parameters is not an easy task.

To overcome the above drawbacks, we introduce two novel fuzzy clustering techniques that use Multiple Kernel Learning to provide an elegant solution for parameter selection. The Fuzzy C-Means with Multiple Kernels algorithm (FCMK) simultaneously finds the optimal partition and the cluster-dependent kernel combination weights that reflect the intrinsic structure of the data. The Relational Fuzzy Clustering with Multiple Kernels (RFCMK) learns the kernel combination weights by optimizing the relational dissimilarities. Consequently, the learned kernel combination weights reflect the relative density, size, and position of each cluster with respect to the other clusters.

We also extended FCMK and RFCMK to the semi-supervised paradigms. We show that the incorporation of prior knowledge in the unsupervised clustering task in the form of a small set of constraints on which instances should or should not reside in the same cluster, guides the unsupervised approaches to a better partitioning of the data and avoid local minima, especially for high dimensional real world data.

All of the proposed algorithms are optimized iteratively by dynamically updating the partition and the kernel combination weights in each iteration. This makes these algorithms simple and fast. Moreover, our algorithms are formulated to work on both vector and relational data. This makes them applicable to data where objects cannot be represented by vectors or when clusters of similar objects cannot be represented efficiently by a single prototype.

We also introduced two relational fuzzy clustering with multiple kernel algorithms for large data to deal with the scalability issue of RFCMK. The random sample and extend RFCMK (rseRFCMK) computes cluster prototypes from a smaller sample of randomly selected objects, and then extends the partition to the remainder of the data. The single pass RFCMK (spRFCMK) sequentially loads manageable sized chunks, clustering the chunks in a single pass, and then combining the results from each chunk.

Our extensive experiments show that RFCMK and SS-RFCMK outperform existing algorithms. In particular, we show that when data include clusters with various intrinsic structures and densities, learning kernel weights that vary over clusters is crucial in obtaining a good partition.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

CHAPTER 1

INTRODUCTION

Clustering is an unsupervised data exploration approach that aims to organize a collection of data items into clusters, such that items within one cluster are more "similar" to each other than they are to items in the other clusters. This notion of similarity can be expressed in very different ways according to domain specific assumptions and to prior knowledge of the problem [62]. Some methods only require the evaluation of pairwise (dis)similarities between data items. Other methods explore the rich representation of the data (e.g. vectorial) that let us define prototypes, data distributions, multidimensional intervals to complement information provided by the computed dissimilarities. While imposing less restrictions on the data, the former methods usually have a higher computational complexity.

Research on unsupervised learning has led to the design and analysis of a wide spectrum of algorithms, including k-means clustering [87], mixture models [87], spectral clustering [118], locality-sensitive hashing [4], and maximum margin clustering [69, 65]. Most of these approaches perform hard clustering. That is, they assign each item to a single cluster. This works well when clustering compact and well-separated groups of data, but in many real-world situations, clusters overlap. Thus, for items that belong to two or more clusters, it may be more appropriate to assign them with gradual memberships to avoid coarse-grained assignments of data [38]. This class of clustering methods is called soft – or fuzzy – clustering.

The Fuzzy C-Means (FCM) [35, 15] is one of the earliest fuzzy clustering methods. In most cases, it is more flexible than the corresponding hard clustering algorithms. Given

the L_2-norm distance in the observation space, it has been shown that while FCM is effective for spherical clusters, it does not perform well for more general clusters.

Recently, kernel-based clustering has been proposed to perform clustering in a typically higher-dimensional feature space spanned by embedding maps and corresponding kernel functions [44]. The FCM algorithm has also been extended to the Kernel Fuzzy C-Means algorithm [128], which yields better performance. However, for such kernel-based methods, a crucial step is the combination or selection of the appropriate kernels among an extensive range of possibilities. This step is often heavily influenced by prior knowledge about the data and by the patterns we expect to discover [105]. Unfortunately, it is often unclear which kernels are more suitable for a particular task [9, 45].

The problem is aggravated for many real-world clustering applications, in which the distributions of the different clusters in the feature space exhibit large variations. Thus, in the absence of a priori knowledge, a single kernel selected from a predefined group is sometimes insufficient to represent the data. Instead of a single fixed kernel, multiple kernels may be used. Recent developments in multiple kernel learning [10, 99, 100] have shown that the construction of a kernel from a number of basis kernels allows for more flexible encoding of domain knowledge from different sources. One way is to construct the kernel from an unweighted sum of kernel functions. Using an unweighted sum gives equal preference to all kernels and this may not be ideal. For instance, these kernel functions are often not equally relevant to the different clusters. Some are less important than others and some maybe completely irrelevant. A better strategy is to learn a convex combination; this also allows extracting information from the weights assigned to kernels. However, most of the previous multiple kernel learning approaches have focused on supervised [10, 100] and semi-supervised learning [72]. An exception to extend multiple kernels to unsupervised learning was proposed by Zhao et al. [9], which is based on maximum margin clustering. However, their method is only designed for hard clustering. Huang et al. [57] designed a multiple kernel fuzzy clustering algorithm which uses one set of global kernel weights for all clusters. Therefore, their approach is not appropriate for clusters of various densities.

Clustering large and high dimensional data collections is a challenging task. The problem is more complex if, in addition to clustering, one is also interested in learning cluster

dependent kernel weights. One possible solution to alleviate this problem is to use partial supervision to guide the search process and narrow the space of possible solutions. Recently, semi-supervised clustering has emerged as a new research direction in machine learning to improve the performance of unsupervised learning using some supervision information. This additional information is usually available in the form of hints [5], constraints [94, 98, 2], or labels [1]. Supervision in the form of constraints is more practical than providing class labels. This is because in many real world applications, the true class labels may not be known, and it is much easier to specify whether pairs of points should belong to the same or different clusters.

Existing semi-supervised clustering algorithms can be grouped into two main categories [93]: search and similarity based approaches. In similarity-based approaches, the similarity metric used to retrieve clusters is first trained to satisfy the constraints in the supervised data. In search-based approaches, the clustering algorithm is modified so as to allow the constraints to "steer" the clustering process towards an appropriate partition. This is usually done by modifying the objective function so that it includes the available supervision provided as pairwise constraints or class labels [82, 14]. These two families of semi-supervised clustering methods rely on slightly different assumptions. Search-based methods consider that the similarities between data items provide relatively reliable information regarding the target categorization, but the algorithm needs some help in order to find the most relevant clusters. Similarity-adapting methods assume that the initial similarity measure has to be significantly modified (at a local or a more global scale) by the supervision in order to correctly reflect the target categorization. While similarity-adapting methods appear to apply to a wider range of situations, they need either significantly more supervision (which can be an unacceptable burden for the user) or specific strong assumptions regarding the target similarity measure (which can be a strong limitation in their domain of application).

The major contributions of this dissertation consist of the design, implementation, and analysis of four clustering algorithms that address the issues raised in the previous discussion. First, the incorporation of multiple kernels and automatic adjustment of the kernel weights in each cluster make the choice of the kernel less crucial and allow better characterization and adaptability to each individual cluster. Second, the fuzzy memberships allow the algorithms to deal with overlapping clusters, and provide a richer description

of the data by distinguishing between core and boundary points of the cluster. Third, the learned kernel combination weights help in identifying clusters of different sizes, shapes, and densities and can be used in subsequent steps to provide better cluster assignment. This dissertation brings several contributions to the fields of kernel methods in machine learning, and computer vision. In the following, we first outline the proposed algorithms. Then, we provide the organization of the remaining of the thesis.

1.1 Fuzzy C-Means with Multiple Kernels

We introduce a new fuzzy prototype-based clustering technique, Fuzzy C-Means with Multiple Kernels (FCMK). This approach simultaneously finds the best degrees of membership and the optimal cluster-dependent kernel weights for a non-negative combination of a set of multi-resolution Gaussian kernels. The incorporation of multiple kernels and the automatic adjustment of kernel weights render FCMK more immune to unreliable kernels. FCMK also makes combining kernels more practical since appropriate weights are assigned automatically. Effective kernels tend to contribute more to the clustering and therefore improve results. Compared to Zhao et al.'s work [9], our approach provides the following advantages. First, our method does not require explicit evaluation in the feature space but conducts only kernel-based evaluations. Thus our method is more suitable for relational data. Second, FCMK is easy to implement. As mentioned by Zhao et al. [9], their formulation leads to a non-convex integer optimization problem which is much more difficult to solve. Finally, FCMK yields fuzzy (soft) clustering results which are more appropriate when clusters have significant overlap, while the Zhao et al. method is designed for crisp clustering.

1.2 Relational Fuzzy C-Means with Multiple Kernels

The Relational Fuzzy C-Means with Multiple Kernels (RFCMK) is another algorithm we developed that learns convex combination of kernel functions with cluster dependent kernel weights while seeking compact clusters. Compared to the first proposed algorithm (FCMK), this algorithm is applicable when data cannot be represented by feature vectors and only the degree to which pairs of objects in the data are related is available. Moreover, even if the data can be represented by feature vectors, the RFCMK is more practical

when similar objects cannot be represented efficiently by a single prototype. Similarly to the FCMK, the RFCMK constructs the kernel from a number of multi-resolution Gaussian kernels and learns a resolution-specific weight for each kernel function in each cluster. This allows better characterization and adaptivity to each individual cluster. RFCMK minimizes one objective function for both the optimal partition and for the cluster dependent kernel weights. This optimization is performed iteratively by dynamically updating the partition and the local measure in each iteration. For instance, the multiple kernel learning task takes advantage of the unlabeled data and reciprocally, the categorization task takes advantage of the local learned combination of kernels.

To the best of our knowledge, FCMK and RFCMK are the first fuzzy algorithms with a cluster-dependent multiple kernel learning setting in an unsupervised way. This is a major contribution to Gaussian-based clustering approaches such as kernel and spectral clustering methods that suffer from their sensitivity to the choice of the scaling parameter.

1.3 Semi-Supervised Fuzzy C-Means with Multiple Kernels

The Semi-Supervised Fuzzy C-Means with Multiple kernels algorithm (SS-FCMK) is an extension of the FCMK. In order to guide FCMK to a better partitioning of the data and avoid local minima, especially for high dimensional real world data, we incorporate prior knowledge in the unsupervised clustering task in the form of a small set of constraints on which instances should or should not reside in the same cluster.

1.4 Semi-Supervised Relational Fuzzy C-Means with Multiple Kernels

The Semi-Supervised Relational Fuzzy C-Means with Multiple Kernels (SS-RFCMK) is an extension of the RFCMK algorithm. It uses side-information in the form of a small set of constraints on which instances should or should not reside in the same cluster. This is achieved by combining constraint-based methods that guide the clustering algorithm towards a better grouping of the data and local distance measure learning methods that adapt the underlying dissimilarity measure used by the clustering algorithm.

1.5 Relational Fuzzy C-Means with Multiple Kernels for Large Data

The random and extend RFCMK (rseRFCMK) and the single pass RFCMK (spRFCMK) are extensions of the RFCMK algorithm to Very Large data (data that we cannot load into our computer's working memory). Currently, the RFCMK does not scale well to large data and requires all data to be clustered to be available in memory. The rseFCMK is based on sampling followed by non-iterative extension. On the other hand, the spRFCMK is an incremental technique that makes one sequential pass through subsets of the data.
The proposed algorithms are compared to existing methods using synthetic and real data sets. We provide detailed analysis of the results and justify their better performance.

The remainder of the dissertation is organized as follows: Chapter 2 provides a literature review of related work relevant to our proposed methods, particularly, prototype-based clustering algorithms, relational clustering algorithms, kernel-based clustering, a semi-supervised learning paradigm and large data clustering. Chapter 3 presents our new unsupervised prototype-based clustering approach in observation space, FCMK. Chapter 4 describes our new relational data clustering method RFCMK. In chapter 5, we present the new semi-supervised prototype-based and relational clustering approaches, SS-FCMK and SS-RFCMK. In chapter 6, we extend RFCMK algorithm for large data. In chapter 7, we describe the experiments conducted to validate the proposed algorithms. The performance of our approaches is compared to several state-of-the-art clustering methods. Finally, Chapter 8 outlines our conclusions and potential future work that could be researched in order to expand the functionality of the proposed algorithms.

CHAPTER 2

LITERATURE SURVEY

Numerous clustering approaches exist, most of which can be divided into three categories: hierarchical, density-based, and partitional clustering. Hierarchical clustering methods [96, 32, 129] partition data by obtaining a nested sequence based on a graphical representation known as a dendrogram. Density-based approaches [70, 54, 42, 126, 120] use local properties of the data objects for grouping purposes. Partitional, also known as prototype-based, methods minimize an objective function and create a single partition [15, 74, 40, 64, 78]. Prototype-based clustering methods provide several advantages when compared to the other methods. In this approach, points are allowed to dynamically shift from one cluster to another. They also provide the ability to incorporate knowledge obtained about cluster shapes and sizes in conjunction with appropriate prototypes and distance measures in their objective functions. While each object is represented by a feature vector in object data representation, for relational representation, only the information of how two objects are related is available. The latter representation is more general in the sense that it can be applied when only the degree of (dis)similarity between objects is available or when groups of similar objects cannot be represented efficiently by a single prototype.

In this chapter, we provide an overview of several clustering methodologies and algorithms relevant to our proposed approaches. The subsequent sections of this chapter are arranged as follows: Section 2.1 reviews a number of prototype-based clustering methods. Methods dedicated to relational clustering are covered in Section 2.2. An overview of kernel-based approaches is given in Section 2.3. In section 2.4, we introduce some

preliminary methods of traditional Multiple Kernel Learning (MKL) in a supervised learning setting. Section 2.5 presents an overview of semi-supervised clustering, which is a search-based approach to partitioning the data where user-provided constraints or labels are used to guide the clustering process. Finally, Section 2.6 presents a digest of clustering in Very Large data (data that cannot be loaded into the computer's working memory).

2.1 Prototype-based clustering

Prototype-based clustering methods represent similar objects efficiently by a single prototype (e.g. center or center and covariance matrix). Moreover, they require an explicit expression of the feature vector of each sample. Prototype-based clustering attempts to find an optimal partition of a data set by minimizing an objective function. The assignment criterion can be described as either hard (crisp) or soft (fuzzy) depending on whether each point belongs to one cluster exclusively or to multiple clusters with varying degrees. In general, fuzzy algorithms perform better than crisp because of their reduced tendency to get trapped in local minima, and their ability to provide a better description of the data [15].

Let $X = \{x_j \in \mathbb{R}^p | j = 1, ..., N\}$ be a set of N feature vectors in a p-dimensional feature space. Let $V = (v_1, ..., v_C)$ represents a C-tuple of prototypes each of which characterizes one of the C clusters. Each v_i consists of a p-dimensional column vector $v_i = [v_{i1}, v_{i2}, ..., v_{ip}]^t$. Let μ_{ij} represent the membership of x_j in cluster i. For the crisp case, the CxN binary C-partition $U = [\mu_{ij}]$, satisfies:

$$\begin{cases} \mu_{ij} \in \{0, 1\}, & \forall i, j \\ 0 < \sum_{j=1}^{N} \mu_{ij} < 1 & \forall i \\ \sum_{i=1}^{C} \mu_{ij} = 1 & \forall j \end{cases} \tag{2.1}$$

for the fuzzy case, U satisfies [43]:

$$\begin{cases} \mu_{ij} \in [0,1] & \forall i,j \\ 0 < \sum_{j=1}^{N} \mu_{ij} < 1 & \forall i \\ \sum_{i=1}^{C} \mu_{ij} = 1 & \forall j \end{cases} \tag{2.2}$$

2.1.1 The K-Means algorithm

In [74], the author describes the K-Means algorithm, one of the earliest unsupervised algorithms for solving the well known clustering problem, partitioning N feature vectors into C clusters. The process is based on iteratively minimizing an objective function known as the vector quantization error, or distortion. In particular, it minimizes

$$J(U,V) = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij} d^2(x_j, v_i) \tag{2.3}$$

subject to the constraints in (2.1).

Minimizing the objective function in (2.3) subject to the constraint in (2.1), is a non-trivial constrained nonlinear optimization problem with no analytical solution. Therefore, an alternating optimization scheme, i.e. alternatively optimizing one set of parameters while the other set of parameters are considered as fixed, is a common approach to optimize (2.3). The first step of the algorithm fixes the cluster centers v_i and assigns points to the nearest cluster based on a given distance $d(x_j, v_i)$. Typically, the Euclidean distance

$$d^2(x_j, v_i) = d_{ij}^2 = ||x_j - v_i||^2 \tag{2.4}$$

is used in the K-Means algorithm. The next step fixes the memberships, μ_{ij}, and (2.3) is optimized with respect to the clusters centroids. This yields an updated equation for the cluster centers:

$$v_i = \frac{\sum_{j=1}^{N} \mu_{ij} x_j}{\sum_{j=1}^{N} \mu_{ij}} \tag{2.5}$$

The K-Means algorithm is summarized in Algorithm 2.1.

Algorithm 2.1 The K-Means algorithm

1: Fix the number of clusters C;
2: Initialize the cluster centroids v_i;
3: **repeat**
4: Assign each object x_j to the nearest cluster;
5: Update the centroids v_i using (2.5);
6: **until** (centers stabilize).

2.1.2 The Fuzzy C-Means algorithm

The FCM algorithm is a modification of the K-Means algorithm that changes the assignment paradigm from crisp to fuzzy [15]. The FCM aims to minimize the following objective function:

$$J(U,V) = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}^{m} d^2(x_j, v_i) \tag{2.6}$$

subject to the constraints in (2.2). In (2.6), $m \in (1,\infty)$ is a weighting exponent (called the fuzzifier) that controls the fuzziness of the partition, and $d(x_j, v_i)$ is the distance from feature point x_j to prototype v_i.

Minimization of (2.6) with respect to the prototype parameters V is dependent on the distance measure. In the initial formulation, the Euclidean distance given by (2.4) was used. It can be shown [15] that the update equations for v_i and μ_{ij} are

$$\mu_{ij} = \frac{(\frac{1}{d^2(x_j,v_i)})^{\frac{1}{m-1}}}{\sum_{q=1}^{C}(\frac{1}{d^2(x_j,v_q)})^{\frac{1}{m-1}}}, \tag{2.7}$$

and

$$v_i = \frac{\sum_{j=1}^{N} \mu_{ij}^{m} x_j}{\sum_{j=1}^{N} \mu_{ij}^{m}} \tag{2.8}$$

The steps of the FCM are outlined in Algorithm 2.2

Like its hard counterpart, the FCM also suffers from the sensitivity to noise and outliers as well as the initial partition. Numerous FCM variants and other fuzzy clustering algorithms have appeared as a result of intensive investigation of the distance measure functions, the effect of the fuzzifier m, the optimization approaches for fuzzy partition, and solutions to overcome the drawbacks of FCM [67, 56].

Algorithm 2.2 The Fuzzy C-Means algorithm

1: Fix $m \in [1, \infty)$;
2: Fix the number of clusters C;
3: Initialize the cluster centers v_i;
4: **repeat**
5: Update the fuzzy memberships μ_{ij} using (2.7);
6: Update the cluster centers v_i using (2.8);
7: **until** no change in μ_{ij}.

2.1.3 The Gustafson-Kessel clustering algorithm

All prototype-based clustering algorithms require the use of some form of similarity measure, typically calculated using a distance measure. The K-Means and FCM algorithms work well only for spherically shaped clusters since the distances from data points to the cluster centers are based on the Euclidean distance (2.4). However, in many applications, clusters, even within the same data set, can have different geometric shapes. In [48], Gustafson and Kessel proposed modifying the FCM algorithm to identify clusters with various shapes. Instead of the Euclidean distance, the authors used an adaptive A-Norm distance given by:

$$d^2(x_j, v_i) = ||x_j - v_i||^2_{A_i} = (x_j - v_i)^T A_i (x_j - v_i) \tag{2.9}$$

subject to

$$det(A_i) = \rho_i \; (constant) \; \forall i. \tag{2.10}$$

where A_i is a local norm-inducing matrix that can be adapted to the local topological structure of each cluster i. For instance, fixing the determinant and varying A allows the algorithm to search for a cluster shape that fits the data while preserving the volume of the cluster. Thus, the GK algorithm is able to find ellipsoidal clusters of different sizes and orientations.

The objective function for the GK algorithm is the same as the FCM (2.6), and minimization yields the same equations for updating the centers (2.8) and the memberships (2.7). To optimize the objective function in (2.6) with respect to A subject to the constraints in (2.2) and (2.10), the authors showed that for each matrix A_i, A_i is a local minimum of (2.6) if

$$A_i^* = (\rho_i |\Sigma_i|)^{1/p} \Sigma_i^{-1}, \; for \; 1 < i < C \tag{2.11}$$

where

$$\Sigma_i = \frac{\sum_{j=1}^{N} \mu_{ij}^m (x_j - v_i)(x_j - v_i)^T}{\sum_{j=1}^{N} \mu_{ij}^m}, \tag{2.12}$$

is the fuzzy covariance matrix. In (2.11), p is the dimensionality of the feature vector x_j.

The GK algorithm is summarized in Algorithm 2.3.

Algorithm 2.3 The Gustafson-Kessel algorithm

1: Fix $m \in [1, \infty)$ and $\rho_i \in [1, \infty)$;
2: Fix the number of clusters C;
3: Initialize the Fuzzy memberships μ_{ij};
4: **repeat**
5: Update the cluster centers v_i using (2.8);
6: Update the local norm-inducing matrices A_i using (2.12) and (2.11);
7: Update the distance using (2.9)
8: Update the fuzzy memberships μ_{ij} using (2.7)
9: **until** (no change in μ_{ij}).

The FCM algorithm has been generalized further to find clusters of different shapes [59]. For instance, in [59] Bezdek et al. proposed the Fuzzy C-Varieties (FCV) algorithm to detect structures of data when all clusters are r-dimensional linear varieties, where r is less than the dimension of the data object. This is achieved by replacing the Euclidean distance in the FCM objective function in (2.6) by the sum of the Euclidean distance and the scaled Euclidean distance mapped to the r principle scatter directions (r longest axes) of the data objects.

2.2 Relational clustering algorithms

While for object data representation, each object is represented by a feature vector, for relational representation, only the information of how pairs of objects are related is available. The latter representation is more general in the sense that it is applicable when the objects to be clustered cannot be represented by numerical features. It is also more practical when the distance measure does not have a closed-form solution, or when groups of similar objects cannot be represented efficiently by a single prototype.

Although clustering of object data has been an active field of research, clustering of relational data has received much less attention. This is despite the fact that several applications would benefit tremendously from relational clustering algorithms. For instance, in several applications, the most effective (dis)similarity measures do not have a closed form expression. Thus, these measures could not be used in object-based algorithms. Examples of these measures include the earth mover distance (EMD) [121, 89], and the integrated region matching distance (IRM) [60]. Similarly, in web data mining and web user profiling, effective similarity measures take into account the URL path traversed [86], and these similarities could not be integrated into object-based clustering methods.

There are several relational clustering algorithms in the literature. One of the well-known relational algorithms is the Sequential Agglomerative Hierarchical Non-overlapping (SHAN) algorithm [107]. When the clusters are overlapping, as is the case in most real-world applications, fuzzy clustering methods are more appropriate. Examples of fuzzy relational clustering methods include the Ruspini algorithm [90] and the Relational Fuzzy C-Means algorithm (RFCM) [84]. In [84], Hathaway et al. reformulated the FCM [15] objective function to adapt it to relational data by eliminating the prototypes from the FCM objective function. They also proposed the Non-Euclidean Relational Fuzzy (NERF) C-Means algorithm [52]. NERF C-Means modifies the RFCM in order to deal with non-Euclidean relational distances. The authors in [80] extended RFCM to the Relational Fuzzy C-Maximal Density estimator algorithm (RFC-MDE). RFC-MDE is robust and learns a local density with respect to each cluster.

A more detailed description of RFCM and NERF is provided in the following subsection.

2.2.1 The Relational Fuzzy C-Means algorithm

In [84], Hathaway et al. reformulated the Fuzzy C-Means (FCM) [15] objective function to adapt it to relational data by eliminating the prototypes from the FCM objective function defined in (2.6). The Relational FCM (RFCM) algorithm minimizes

$$J(U) = \sum_{i=1}^{N} \frac{\sum_{j=1}^{N} \sum_{k=1}^{N} \mu_{ij}^{m} \mu_{ik}^{m} r_{jk}}{2 \sum_{k=1}^{N} \mu_{ik}^{m}}, \tag{2.13}$$

subject to the membership constraint in (2.2).

Unlike the object-based FCM objective function which involves dissimilarity of the objects to a cluster center, the objective function in (2.13) includes only the dissimilarities r_{jk} between pairs of objects. The dissimilarities r_{jk} could be provided or computed from the object data using

$$r_{jk} = \|x_j - x_k\|^2. \tag{2.14}$$

In [84], it was shown that the minimization of the FCM and RFCM objective functions is equivalent. To derive the update equations for the RFCM, Hathaway et al. [84] proved that the squared Euclidean distance, $d_{ik}^2 = \| c_i - x_k \|^2$, from feature vector x_k and the centroid of the i^{th} cluster c_i, can be written in terms of the relation matrix $R = [r_{jk}]$ as

$$d_{ik}^2 = \left(Rv_i\right)_k - \frac{v_i^t R v_i}{2} \tag{2.15}$$

where v_i is the membership vector defined by

$$v_i = \frac{\left(\mu_{i1}^m, \ldots, \mu_{iN}^m\right)^t}{\sum_{j=1}^N \mu_{ij}^m} \tag{2.16}$$

Equation (2.15) allows the computation of the distance between the data points and cluster prototypes in each iteration with no explicit cluster centroid. It uses only the relational data, R, and the set of initial fuzzy memberships. Once the implicit distance values, d_{ik}^2, have been computed using (2.15), the fuzzy memberships are updated as in the standard FCM using

$$\mu_{ik} = \frac{1}{\sum_{t=1}^C \left(d_{ik}^2 / d_{tk}^2\right)^{\frac{1}{m-1}}}. \tag{2.17}$$

The RFCM objective function in (2.13) is optimized by alternating between the update equations in (2.15) and (2.17) until the membership values do not change.

RFCM was formulated to cluster data that is available in the form of relational matrix without explicit knowledge of the coordinates of the objects in the feature space. It is expected to perform in an equivalent way to the FCM provided that the relation matrix, R, is Euclidean i.e. (2.14) is satisfied. When this is not the case, the relational dual of the FCM can fail mainly because some of the distances computed using (2.15) may

be negative. To overcome this restriction, Hathaway and Bezdek proposed the Non-Euclidean Relational Fuzzy (NERF) C-Means algorithm [52]. NERF C-Means modifies the RFCM by adding a step that uses the β-spread transform to convert a non-Euclidean matrix R into an Euclidean matrix R_β as follows

$$R_\beta = R + \beta(M - I). \tag{2.18}$$

In (2.18), β denotes a suitably chosen scalar, $I \in \mathbb{R}^{N \times N}$ is the identity matrix, and M is an NxN matrix whose entries are all equal to one. In [52], the authors suggested that the distance d_{ik}^2 should be checked in every iteration for negativity, which indicates a non-Euclidean relation matrix. In that case, the β-spread transform should be applied with a suitable value of β to make d_{ik}^2 positive. A lower bound for the necessary shift, $\Delta\beta$, that is needed to make the distances positive was derived in [52] to be

$$\Delta\beta = \max_{i,k} \left\{ -2d_{ik}^2 / \|v_i - e_k\|^2 \right\}, \tag{2.19}$$

where e_k denotes the k^{th} column of the identity matrix. The steps of the NERF C-Means are outlined in Algorithm 2.4.

Algorithm 2.4 The Non-Euclidean Relational Fuzzy (NERF) C-Means algorithm

1: Fix the number of clusters C and $m \in [1, \infty)$;
2: Initialize $\beta = 0$;
3: Initialize the fuzzy partition matrix U;
4: **repeat**
5: Compute R_β using (2.18);
6: Compute the membership vector v_i using (2.16);
7: Compute the distances using (2.15);
8: **if** $d_{ik}^2 < 0$ for any i, k **then**
9: Compute $\Delta\beta$ using (2.19);
10: $d_{ik}^2 = d_{ik}^2 + \left(\Delta\beta/2\right) * \|v_i - e_k\|$;
11: $\beta = \beta + \Delta\beta$;
12: **end if**
13: Update the fuzzy memberships μ_{ij} using (2.17);
14: **until** (fuzzy membership do not change).

The relational clustering algorithm described above clusters the data in its original feature space. However, in many real applications, categories may be defined better in a transformed feature space. Kernel based methods identify clusters in a transformed space.

2.3 Kernel clustering

Kernel-based learning algorithms [41, 102, 115] are based on Cover's theorem. By non-linearly transforming a set of complex and non-linearly separable patterns into a higher-dimensional feature space, it is possible to separate these patterns linearly [53]. The difficulty of curse of dimensionality can be overcame by the kernel trick, arising from Mercer's theorem [53]. By designing and calculating an inner-product kernel, we can avoid the time-consuming, sometimes even infeasible process to explicitly describe the nonlinear mapping $\Phi : X \mapsto \mathcal{F}$ from the input space X to a high dimensional feature space $\mathcal{F}$ and compute the corresponding points in the transformed space.

One of the most relevant aspects in kernel based learning is that it is possible to compute the Euclidean distances in $\mathcal{F}$ without knowing explicitly Φ. This can be done using the so called distance kernel trick:

$$\begin{aligned} ||\Phi(x_i)-\Phi(x_j)||^2 &= (\Phi(x_i)-\Phi(x_j))^t.(\Phi(x_i)-\Phi(x_j)) \\ &= \left(\Phi(x_i)^t\Phi(x_i)\right)+\left(\Phi(x_j)^t\Phi(x_j)\right)-2\left(\Phi(x_i)^t\Phi(x_j)\right) \\ &= K(x_i,x_i)+K(x_j,x_j)-2K(x_i,x_j) \end{aligned} \tag{2.20}$$

Thus, the computation of distances of vectors in feature space is just a function of the input vectors. In fact, every algorithm in which input vectors appear only in dot products with other input vectors can be kernelized.

In (2.20), $K(x_i,x_j) = \Phi(x_i)^t\Phi(x_j)$ is the Mercer kernel. It is a symmetric function $K : X \times X \longrightarrow \mathbb{R}$ and satisfies

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j K(x_i,x_j) \geq 0 \quad \forall N \geq 2, \tag{2.21}$$

where $c_r \in \mathbb{R}\ \forall r = 1,\ldots,N$. Examples of Mercer kernels include [116]

- linear

$$K^{(l)}(x_i,x_j) = x_i^t \cdot x_j \tag{2.22}$$

- polynomial of degree p

$$K^{(p)}(x_i,x_j) = (1 + x_i^t \cdot x_j)^p, \quad p \in \mathbb{N} \tag{2.23}$$

- Gaussian

$$K^{(g)}(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \quad \sigma \in \mathbb{R} \tag{2.24}$$

Kernel-based clustering algorithms have the following main advantages.

1. They are more likely to obtain a linearly separable hyperplane in the high-dimensional, or even in an infinite feature space;
2. They can identify clusters with arbitrary shapes;
3. Kernel-based clustering algorithms, like support vector clustering (SVC), have the capability of dealing with noise and outliers;
4. For SVC, there is no requirement for prior knowledge to determine the system topological structure. In [44], the kernel matrix can provide the means to estimate the number of clusters.

The following subsections outline some well-known kernel clustering algorithms.

2.3.1 The kernel K-Means algorithm

In general, the mapping ϕ from the input space X to a high dimensional feature space $\mathcal{F}$ is not known. Thus, it is not possible to compute directly the center of each cluster i, c_i^{ϕ}, in the feature space $\mathcal{F}$ using

$$c_i^{\phi} = \frac{1}{|\Pi_i^{\phi}|} \sum_{x \in \Pi_i^{\phi}} \phi(x_h) \tag{2.25}$$

In (2.25), Π_i^{ϕ} is the set of points x_h belonging to cluster i. Let γ be the indicator matrix in which γ_{ih} is equal to one if x_h belongs to cluster i and zero otherwise. Alternatively, the distances in the feature space could be computed using

$$\begin{aligned} \|\phi(x_j) - c_i^{\phi}\|^2 &= \|\phi(x_j) - \frac{1}{|\Pi_i^{\phi}|} \sum_{x \in \Pi_i^{\phi}} \phi(x_h)\|^2 \\ &= K(x_j, x_j) - 2\sum_h \gamma_{ih} K(x_j, x_h) + \sum_r \sum_s \gamma_{ir}\gamma_{is} K(x_r, x_s) \end{aligned} \tag{2.26}$$

The kernel K-Means algorithm [71] is summarized in Algorithm 2.5.

Algorithm 2.5 The Kernel K-Means algorithm

1: Fix the number of clusters C;
2: Initialize the cluster centers in the feature space $\mathcal{F}$;
3: **repeat**
4: Compute distances between the data points and cluster centers using (2.26) for each cluster;
5: Update the indicator matrix γ by assigning data points to the nearest cluster according to the computed distances.
6: **until** (γ do not change).

2.3.2 The kernelized metric Fuzzy C-Means algorithm

The kernelized metric Fuzzy C-Means [124] minimizes the following objective function:

$$J^{\phi} = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}^{m} \|\phi(x_j) - \phi(c_i)\|^2, \tag{2.27}$$

subject to the constraint in (2.2). In (2.27), $\phi(c_i)$ is the center of cluster i in the feature space, and ϕ is the mapping from the input space X to the feature space $\mathcal{F}$.
Minimization of the function in (2.27) has been proposed only in the case of a Gaussian kernel. The reason is that in this case the derivative with respect to c_i can use the kernel trick:

$$\frac{\partial K(x_j, c_i)}{\partial c_i} = \frac{(x_j, c_i)}{\sigma^2} K(x_j, c_i). \tag{2.28}$$

It can be shown [124] that the update equation for the memberships is

$$\mu_{ij} = \frac{1}{\sum_{h=1}^{C} \left(\frac{1-K(x_j, c_i)}{1-K(x_j, c_h)} \right)^{1/(m-1)}} \tag{2.29}$$

and for the code-vectors is

$$c_i = \frac{\sum_{j=1}^{N} \mu_{ij}^{m} K(x_j, c_i) x_j}{\sum_{j=1}^{N} \mu_{ij}^{m} K(x_j, c_i)}. \tag{2.30}$$

The kernelized metric Fuzzy C-Means algorithm is outlined in Algorithm 2.6.

2.3.3 The kernel Fuzzy C-Means in feature space algorithm

The kernel Fuzzy C-Means in feature space [127] algorithm allows to find a soft linear partitioning of the feature space. This partitioning, back to the input space, results in

Algorithm 2.6 The Kernelized metric Fuzzy C-Means algorithm

1: Fix the number of clusters C and $m \in [1, \infty)$;
2: Initialize the cluster centers c_i;
3: **repeat**
4: Update the fuzzy memberships μ_{ij} using (2.29);
5: Update the cluster centers c_i using (2.30);
6: **until** (μ_{ij} do not change).

a soft nonlinear partitioning of data. The functional to optimize with the probabilistic constraint in (2.2) is

$$J^{\phi} = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^{m} \| \phi(x_j) - c_i^{\phi} \|^2. \tag{2.31}$$

It is possible to rewrite the norm in (2.31) explicitly by using:

$$\begin{aligned} c_i^{\phi} &= \frac{\sum_{j=1}^{N} \mu_{ij}^{m} \phi(x_j)}{\sum_{j=1}^{N} \mu_{ij}^{m}} \\ &= a_i \sum_{j=1}^{N} \mu_{ij}^{m} \phi(x_j) \end{aligned} \tag{2.32}$$

where

$$a_i = \frac{1}{\sum_{j=1}^{N} \mu_{ij}^{m}}. \tag{2.33}$$

This trick allows the derivation of a closed form expression for the membership update equation

$$\mu_{ij} = \frac{1}{\sum_{h=1}^{C} \left[\frac{K_{jj} - 2a_i \sum_{r=1}^{N} \mu_{ir}^{m} K_{jr} + a_i^2 \sum_{r=1}^{N} \sum_{s=1}^{N} \mu_{ir}^{m} \mu_{is}^{m} K_{rs}}{K_{jj} - 2a_h \sum_{r=1}^{N} \mu_{hr}^{m} K_{jr} + a_h^2 \sum_{r=1}^{N} \sum_{s=1}^{N} \mu_{hr}^{m} \mu_{hs}^{m} K_{rs}} \right]^{1/(m-1)}}. \tag{2.34}$$

The kernel Fuzzy C-Means in feature space is outlined in Algorithm 2.7.

Algorithm 2.7 The Kernel Fuzzy C-Means in feature space algorithm

1: Fix the number of clusters C and $m \in [1, \infty)$;
2: Initialize the cluster centers c_i;
3: **repeat**
4: Update the fuzzy memberships μ_{ij} using (2.34);
5: Update the cluster centers c_i using (2.32);
6: **until** (μ_{ij} do not change).

2.3.4 The kernelized Non-Euclidean Relational Fuzzy C-Means algorithm

The kernelized Non-Euclidean Relational Fuzzy (kNERF) C-Means algorithm [52] works directly on relational data and does not require explicit feature representation. In particular, the Gaussian kernelized relational matrix $\widehat{R}$ can be formed directly from the original relational data matrix $R = [r_{jk}]$ using

$$\widehat{R} = 1 - \exp\left(-\frac{r_{jk}^2}{\sigma^2}\right), \tag{2.35}$$

and the Non-Euclidean Relational Fuzzy (NERF) C-Means algorithm is adapted to perform clustering analysis on the kernelized matrix $\widehat{R}$.

Despite the existence of a large number of kernel-based clustering algorithms (e.g. kernel K-Means, kernel FCM, kernel SOM and kernel Neural Gas), the choice of a good kernel function and the adaptation of its parameters to the data remains a challenging task. For instance, in kNERF, a relational Gaussian kernel is used with one global scaling parameter for the entire data. The selection of the scaling parameter is discussed in [88] but there has been no attempt to derive methods to automatically select it. Moreover, a global scaling parameter may not be appropriate when clusters have different shapes, sizes, and densities.

2.4 Multiple Kernel Learning

The crux of kernel methods is the *kernel*, which is in general a function that defines an inner product of any two samples in some induced Hilbert space [105, 111]. By mapping data from an input space to some Reproducing Kernel Hilbert space (RKHS) which can be potentially high-dimensional, traditional linear methods can be extended with reasonable effort to yield considerably better performance. Many empirical studies have shown that the choice of kernel often affects the resulting performance of a kernel method significantly. In fact, inappropriate kernels usually result in sub-optimal or even poor performance when applying kernel methods to solve a real-world problem.

For many real-world situations, it is often not easy to choose an appropriate kernel, which usually requires some domain knowledge that may be difficult for non-expert users. To address such limitations, recent years have witnessed the active research on learning

effective kernels automatically from data [47, 99, 101, 61]. One popular technique for kernel learning is Multiple Kernel Learning (MKL) [47, 99], which aims at learning a linear (or convex) combination of a set of predefined kernels in order to identify a good target kernel for the application. Compared to traditional kernel methods using a single fixed kernel, MKL does exhibit its strength of automated kernel parameter tuning and capability of concatenating heterogeneous data. Over the past few years, MKL has been actively studied, in which a variety of algorithms have been proposed to solve the efficiency of MKL [99, 125], and a number of extended MKL techniques have been proposed to improve the regular MKL method [43, 117, 25, 73, 85].

Despite being studied extensively, existing MKL methods are essentially supervised and require class labels of training data available for the kernel learning task. However, in many unsupervised learning tasks, such as dimension reduction or clustering, class labels may not always be available.

2.4.1 The Supervised Multiple Kernel Learning

In a typical supervised multiple kernel learning task, we are given a collection of N training samples $\mathscr{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $x_i \in \mathbb{R}^p$ is the input feature vector and y_i is the class label of x_i. The conventional multiple kernel learning (MKL) can be formulated into the following optimization task:

$$\min_{K \in \mathscr{K}} \min_{f \in \mathscr{H}_K} \lambda \|f\|_{\mathscr{H}_K} + \sum_{i=1}^{N} \mathscr{L}(y_i f(x_i)), \tag{2.36}$$

where $\mathscr{L}(.)$ denotes some loss function, e.g. the hinge loss function $\mathscr{L}(t) = \max(0, 1-t)$ used for SVM, $\mathscr{H}_K$ is the reproducing kernel Hilbert space associated with kernel K, $\mathscr{K}$ denotes the optimization domain of the candidate kernels, and λ is a regularization parameter.

The above optimization aims to simultaneously identify both the optimal kernel K from domain $\mathscr{K}$ and the optimal prediction function f from the kernel Hilbert space $\mathscr{H}_K$ induced by the optimal kernel K. By the represented theorem [8], the decision function $f(x)$ is in form of a linear expansion of kernel evaluation on training sample x_i, i.e.,

$$f(x) = \sum_{i=1}^{N} \beta_i K(x_i, x),$$

where β_i are the coefficients. In traditional MKL [47], $\mathscr{K}$ is chosen to be a set of convex combination of M predefined base kernels, i.e.,

$$\mathscr{K}_{conv} = \left\{ K(.,.) = \sum_{t=1}^{M} w_t K_t(.,.) : \sum_{t=1}^{M} w_t = 1, w_t \geq 0 \right\}, \tag{2.37}$$

where each candidate kernel K is some combination of the M base kernels $\{K_1, \ldots, K_M\}$ and w_t is the coefficient of the t^{th} base kernel. Based on (2.37), the decision function of regular MKL can be written as:

$$f(x) = \sum_{i=1}^{N} \sum_{t=1}^{M} \beta_i w_t K_t(x_i, x)$$

Although MKL in general can be formulated as a convex optimization task, such an optimization task is usually difficult to solve. In literature, researchers have spent extensive efforts in developing efficient solvers for supervised MKL. Some representative examples include [43, 117, 25, 73]. In addition to the efficiency issue, some recent studies have attempted to overcome some limitations of regular MKL techniques [25]. For example, some studies have addressed the limitation of using linear combination of multiple kernels by exploring more flexible kernel combination methods [43, 117, 25, 61]. Most of these methods essentially follow the same large margin learning framework of SVM. Very recently, Cortes et. al. [22] proposed the two-stage kernel target alignment, which isolates the kernel learning task from SVM. It exhibits the state-of-the-art empirical performance when comparing with other conventional kernel learning techniques.

In MKL, the objective function follows the regularization framework of SVM, i.e., loss and regularization. The norm of f in RKHS as a regularization term penalizes the complexity of SVM effectively as it upper bounds of the Rademacher complexity of the function class induced by a single kernel [13]. When the kernel class varies during the learning, it always introduces more complexity [47, 108, 123].

2.5 Semi-supervised clustering

In most applications, clustering can be a challenging task. For instance, completely unsupervised clustering methods tend to get trapped in local minima due to complex objective functions. Moreover, these algorithms form clusters based solely on the similarity between objects provided by a given similarity measure. Therefore, these methods rely heavily on the choice of the similarity measure and may not provide semantically meaningful clusters. Alternatively, the use of supervised methods may not be an option because labeling all the data could be a very expensive or even an impossible task. For instance, in large data sets even the overall number of classes may not be known.

Recent research into an area known as semi-supervised clustering has been proposed in an attempt to improve the performance of unsupervised learning [91, 92, 64, 77]. Semi-supervised methods are formulated using side information associated with the data in order to guide or adjust the clustering process. Typically, the information is presented in the form of labels [81, 6, 91], constraints [119, 30, 92], or hints [5]. In real world applications, the true labels of the data may not be known. Therefore, it may be more practical to specify which pairs of points should or should not belong to the same cluster. Thus, it is more practical to incorporate the partial supervision in the form of constraints as opposed to class labels. Although semi-supervised methods have been proven to outperform their unsupervised counterparts, the extensiveness of the research into semi-supervised methods is not as vast [33, 17, 77].

The majority of semi-supervised clustering methods can be dichotomized as similarity-adapting [30, 18, 26, 36, 83] or search-based [3, 119, 64, 91, 92, 77]. Since our approach uses a search-based method of semi-supervision, only search-based methods will be outlined in this literature survey.

Search-based semi-supervised clustering methods adapt existing clustering algorithms, using constraints or labels, to bias the search for a semantically more meaningful partition, and to avoid local minima. There are various methods in which this information can be incorporated. Some methods use constraints to perform transitive closure and initialize the clustering process [91]. Other methods incorporate the constraints into the optimization process. In the latter approach, some methods force all constraints to be satisfied during the clustering process [3], while some other methods use modifications

to the objective function, penalizing the process when constraints are left unsatisfied [119].

One way of incorporating the background knowledge is to use pairwise constraints [64, 30, 31, 77]. Generally, these constraints are defined as either must-link or cannot-link [119]. When two points are required to be in the same cluster they form a must-link constraint. In contrast, if the two points are intended to be in different clusters, they form a cannot-link constraint. These constraints are formulated in order to guide the clustering process to find both naturally occurring patterns with a user-defined overtone. The constraint satisfaction criteria can be defined as either being strict or relaxed. If the satisfaction criterion is defined as strict, the constraints are to be unconditionally satisfied [3, 64]. A relaxed criterion, on the other hand, implies that the constraints may or may not be satisfied [76].

2.5.1 The semi-supervised Fuzzy C-Means

In [77], a semi-supervised Fuzzy C-means algorithm (SS-FCM) was proposed. It minimizes the following objective function

$$J = \sum_{i=1}^{C}\sum_{j=1}^{N}\mu_{ij}^2 d_{ij}^2 + \alpha\left(\sum_{(x_j,x_k)\in Ml}\sum_{i=1}^{C}\sum_{l=1,l\neq i}\mu_{ij}\mu_{lk} + \sum_{(x_j,x_k)\in Cl}\sum_{i=1}^{C}\mu_{ij}\mu_{ik}\right) \tag{2.38}$$

under the same constraints (2.2).

In (2.38), Ml is a set of must-link constraints such that $(x_j, x_k) \in Ml$ implies that the objects x_j and x_k must be assigned to the same cluster. Similarly, Cl is a set of cannot-link constraints such that $(x_j, x_k) \in Cl$ implies that the objects x_j and x_k cannot be assigned to the same cluster. The first term in (2.38) is the sum of squared distances to the prototypes weighted by constrained memberships (Fuzzy C-Means objective function). The second component in (2.38) is composed of the cost of violating the pairwise must-link constraints and the cost of violating the pairwise cannot-link constraints. This term is weighted by α, a constant factor that specifies the relative importance of the supervision. The prototypes of the clusters are given by

$$c_i = \frac{\sum_{j=1}^{N}\mu_{ij}^2 x_j}{\sum_{j=1}^{N}\mu_{ij}^2}. \tag{2.39}$$

To minimize (2.38) with respect to U under the constraints (2.2), the authors use Lagrange multipliers and obtain the following updating equations of the memberships values

$$\mu_{ij} = \mu_{ij}^{FCM} + \mu_{ij}^{Constraints} \tag{2.40}$$

where

$$\mu_{ij}^{FCM} = \frac{\frac{1}{d_{ij}^2}}{\sum_{l=1}^{C} \frac{1}{d_{lj}^2}}, \tag{2.41}$$

and

$$\mu_{ij}^{Constraints} = \frac{\alpha}{2d_{ij}^2}\left(\overline{Cv_j} - Cv_{ij}\right). \tag{2.42}$$

In (2.42), $\overline{Cv_j}$ and Cv_{ij} are defined as

$$Cv_{ij} = \sum_{(x_j,x_k)\in Ml} \sum_{l=1,l\neq i} \mu_{lk} + \sum_{(x_j,x_k)\in Cl} \mu_{ik} \tag{2.43}$$

and

$$\overline{Cv_j} = \frac{\sum_{i=1}^{C} \frac{\sum_{(x_j,x_k)\in Ml} \sum_{l=1,l\neq i} \mu_{lk} + \sum_{(x_j,x_k)\in Cl} \mu_{ik}}{d_{ij}^2}}{\sum_{i=1}^{C} \frac{1}{d_{ij}^2}}. \tag{2.44}$$

The α factor should provide a balance between the constrained data and unlabeled patterns. Thus, α is defined as

$$\alpha = \frac{N}{M} \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^2 d_{ij}^2}{\sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^2}, \tag{2.45}$$

where M denotes the number of pairwise constraints.

In (2.45), α is the normalized performance index which is the sum of the constrained squared distances between patterns and prototypes over the sum of the squared membership degrees. For instance, high values mean that we still need supervision to structure data while lower values means that the data are getting allocated to the right "supervised" clusters and we need less guidance than in early stages.

The resulting SS-FCM algorithm is summarized in Algorithm 2.8.

As an attempt to extend SS-FCM to relational data, Frigui et al. [39] proposed a semi-supervised algorithm that clusters and aggregates relational data (SS-CARD). This algorithm not only partitions the data into meaningful clusters using pairwise must-link

Algorithm 2.8 The Semi-Supervised Fuzzy C-Means algorithm

1: Fix the number of clusters C;
2: Initialize the fuzzy partition matrix U;
3: Create Ml and Cl pairwise constraints;
4: **repeat**
5: Compute α using (2.45);
6: Compute prototypes using (2.8);
7: Update memberships using (2.40);
8: **until** (fuzzy memberships do not change)

and cannot-link constraints between few data samples, but also aggregates pairwise distances from multiple relational matrices and learns a relevance weight for each matrix in each cluster.

2.5.2 The semi-supervised kernel C-Means algorithm

In [7], a relational semi-supervised kernel clustering algorithm (SS-Kernel-CMeans) was proposed. It minimizes the following objective function

$$\begin{aligned} J = & \sum_{i=1}^{C} \sum_{(x_j,x_k)\in\pi_i} \frac{\sum_{j=1}^{N}\sum_{k=1}^{N} K_{jk}}{2|\pi_i|} \\ & -\sum_{i=1}^{C} \sum_{(x_j,x_k)\in Ml} \sum_{(x_j,x_k)\in\pi_i} \frac{2\theta_{jk}}{|\pi_i|} + \sum_{i=1}^{C} \sum_{(x_j,x_k)\in Cl} \sum_{(x_j,x_k)\in\pi_i} \frac{2\theta_{jk}}{|\pi_i|} \end{aligned} \tag{2.46}$$

In (2.46), Ml is a set of must-link constraints such that $(x_j, x_k) \in Ml$ implies that the objects x_j and x_k must be assigned to the same cluster. Similarly, Cl is a set of cannot-link constraints such that $(x_j, x_k) \in Cl$ implies that the objects x_j and x_k cannot be assigned to the same cluster.

In (2.46), π_i is the set of points that belong to cluster i, $|\pi_i|$ is its corresponding cardinality, N and C are respectively the number of data objects and the number of clusters, and K_{jk} is the relational Gaussian kernel function. Let Θ be the constraint matrix such that

$$\Theta = \begin{cases} -\theta_{ij} & if\ (x_j, x_k) \in Cl \\ \theta_{ij} & if\ (x_j, x_k) \in Ml \\ 0 & otherwise \end{cases} \tag{2.47}$$

Let e_i be an indicator vector for cluster i. This vector is of length N and $e_i(j) = 0$ if x_j is not in cluster i, and 1 otherwise. The objective in (2.46) can be written as follows

$$J = \sum_{i=1}^{C} \sum_{(x_j, x_k) \in \pi_i} \frac{\sum_{j=1}^{N} \sum_{k=1}^{N} e_j^t (K_{jk} + 2\Theta_{jk}) e_k}{e_j e_k^t} \tag{2.48}$$

By setting $\hat{K} = K_{jk} + 2\Theta_{jk}$, the authors in [7] showed that the SS-Kernel-CMeans objective function in (2.48) is equivalent to the kernel k-means one.

The steps of the semi-supervised kernel C-Means are outlined in Algorithm 2.9.

Algorithm 2.9 The Semi-Supervised Kernel C-Means algorithm

1: Fix the number of clusters C, and the similarity matrix K;
2: Fix the constraint penalty matrix Θ;
3: **repeat**
4: Form the matrix $\hat{K} = K + \Theta$;
5: For each cluster, compute the distances between the data points and the cluster centers using 2.20
6: Update the indicator matrix e_i by assigning data points to the nearest cluster according to the computed distances;
7: **until** (no change in e_i)

The semi-supervised kernel C-Means has several drawbacks. Fist, since it is a crisp version, it is sensitive to initialization and cannot deal with overlapping boundaries. Second, there has been no method suggested to automate the selection of the scaling parameter and this parameter has to be set manually. Finally, it is not clear how to generate the constraint weights θ_{jk}.

2.5.3 The semi-supervised Spectral Learning

Recently, graph-based spectral clustering algorithms [50, 106, 75] have been developing rapidly. In the first step, a similarity graph and its corresponding weighted adjacency matrix are constructed. The fully connected graph is very often used in connection with the Gaussian similarity function. The second step of the algorithm consists of computing the Laplacian. Then the smallest eigenvectors are computed and concatenated in order to constitute the new space feature matrix. The rows of this matrix are then clustered with the C-Means algorithm [74].

In [29], the authors generalize the unsupervised spectral clustering to semi-supervised spectral clustering to take advantage of available supervision information. This algorithm does not have an explicit underlying objective function. It simply injects the pairwise constraints into the affinity matrix before clustering. In fact, since for most similarity functions, the maximum pairwise similarity value is 1, and the minimum similarity is 0, the authors assigned 1 to the affinity matrix entries corresponding to a Must-Link pair of points and 0 to the affinity matrix entries corresponding to a Cannot-Link pair of points. The semi-supervised spectral is outlined in Algorithm 2.10.

Algorithm 2.10 The Semi-Supervised spectral algorithm

1: Fix the number of clusters C;
2: Construct a similarity graph and the weighted adjacency matrix R_{ij}. The fully connected graph is very often used in connection with the Gaussian similarity function

$$R_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right) & \text{if } i \neq j \\ 0 & \text{subject to } x_i \text{feature vector } i \end{cases} \tag{2.49}$$

3: Assign 1 to the affinity matrix entries corresponding to a Must-Link and 0 to the affinity matrix entries corresponding to a Cannot-Link;
4: Compute the degree matrix

$$D = diag\left(\sum_j R_{ij}\right) \tag{2.50}$$

5: Compute the Laplacian;
6: Compute the first C eigenvectors $v_1, \ldots, v_C$ and let V be the matrix containing this vectors;
7: **for** $i = 1$ to N **do**
8: Let y_i be the vector corresponding to the i^{th} row of V;
9: Cluster the points $\{y_i\}_{i=1,\ldots,N}$ with the K-Means algorithm into $C_1, \ldots, C_C$ clusters.
10: **end for**

2.6 Clustering in Very Large Data

Much research has been done on clustering in VL data. Methods can be roughly categorized into three types of algorithms. First, sampling methods compute cluster centers from a smaller sample of (often randomly) selected objects. Popular sampling based methods include CLARA [66], CURE [97], and the coreset algorithms [51]. Second,

single-pass algorithms sequentially load small groups of the data, clustering these manageable chunks in a single pass, and then combining the results from each chunk. Representative algorithms include incremental clustering [23, 37] and divide-and-conquer approaches [21, 95]. Finally, data transformation algorithms alter the structure of the data itself so that it is more efficiently accessed. The data often take the form of graph-like structures. Well-known algorithms of this type include BIRCH [112] and CLARANS [79]. While all these algorithms perform their job well, they all produce crisp partitions. One of the most well-known method for fuzzy clustering of VL data is the generalized extensible fast FCM (geFFCM) [16]. This algorithm uses statistics-based progressive sampling to produce a reduced dataset that is large enough to capture the overall nature of the data. It then clusters this reduced dataset and non iteratively extends the partition to the full dataset. However, the sampling method used in geFFCM can be inefficient and, in some cases, the data reduction is not sufficient for VL data. Hence, the geFFCM algorithm was adapted into a simple random sampling plus extension FCM (rseFCM) algorithm in [113]. Other leading algorithms include single-pass FCM (spFCM) [68] and online FCM (oFCM) [27], which are incremental algorithms to compute an approximate FCM solution. The bit-reduced FCM (brFCM) [67] algorithm uses a binning strategy for data reduction. A kernel-based strategy which is called approximate kernel FCM (akFCM) was developed in [110, 109], which relies on a numerical approximation that uses sampled rows of the kernel matrix to estimate the solution to a c-means problem. Other methods that generate fuzzy partitions include the fast FCM (FFCM) developed in [104], where FCM is applied to larger and larger nested samples until there is little change in the solution; and the multistage random FCM proposed in [28], which combines FFCM with a final literal run of FCM on the full dataset. Both these schemes are more in the spirit of acceleration, rather than scalability, as they both contain a final run on the full dataset.

Among various existing algorithms proposed for VL data clustering, several studies are devoted to examining kernel techniques in incremental learning settings [34, 63, 110, 113]. However, most of the existing kernel based incremental learning algorithms assume that the kernel function is given a priori, significantly limiting their applications to real world problems.

CHAPTER 3

PROTOTYPE-BASED CLUSTERING WITH MULTIPLE KERNELS

For real-world clustering tasks, the input data rarely correspond to well defined clusters due to the highly complex data structure and noise. Kernel based algorithms [49, 44, 24, 19, 71] attempt to improve the structure of the data by mapping them into high-dimensional Hilbert space $\mathscr{H}$ via a mapping function $\phi(.)$. An attractive property of kernelized algorithms is that only the so-called Gram matrix $G = [K(x_i, x_j)]$, rather than explicit mapping functions, where $K(x_i, x_j)$ corresponds to the inner product of $\phi(x_i)$ and $\phi(x_j)$ in $\mathscr{H}$ need to be provided. However, a good choice of the kernel function is imperative to the success of such kernel-based methods. This choice is often heavily influenced by prior knowledge about the data and by the patterns we expect to discover [105]. Unfortunately, it is often unclear which kernel is the most suitable for a particular task [9, 45]. The problem is aggravated for many real-world applications, in which there are multiple potentially useful kernels. For instance, in applications involving data clustering, it may not be possible to find one optimal kernel function for the entire data set where large variations between the distributions of the different clusters in the feature space are very common.

Instead of a single fixed kernel, multiple kernels may be used. Recent studies have sparked interest in multiple kernel learning (MKL) [46, 10, 100, 122], i.e., utilizing multiple heterogeneous kernels simultaneously. MKL provides an elegant solution for

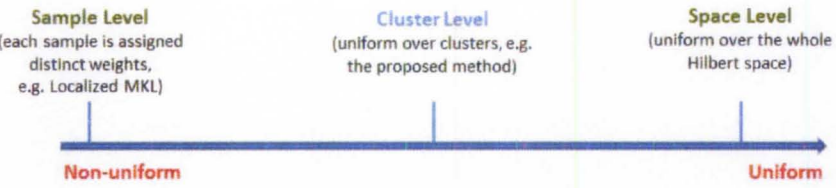


Figure 3.1: Illustration of the evolution of various MKL methods.

parameter selection. In Figure (3.1), we plot the evolution line of various MKL algorithms. Most existing MKL methods assume that kernel weights remain constant for all data (i.e., space-level), while algorithms like Localized MKL [72] seek kernel weights that are data-dependent and locally smooth (i.e., sample-level). Although sample-level non-uniform methods give the largest flexibility, in practice relaxations are typically introduced to enhance tractability.

To overcome the above drawbacks and build MKL algorithms with a better tradeoff between flexibility and tractability, we propose a novel, practical MKL method on the cluster-level. Our main observation lies in that data from the same cluster tend to manifest similar properties, thus the intra-cluster weights can be approximately uniform while kernel weights are allowed to vary over clusters. The proposed Fuzzy Clustering with Multiple Kernels (FCMK) algorithm[1] strives to find a good partitioning of the data into meaningful clusters and the optimal kernel-induced feature map in a completely unsupervised way. FCMK constructs the kernel from a number of Gaussian kernels and learns a resolution weight for each kernel function in each cluster. The incorporation of multiple kernels and the automatic adjustment of kernel weights in each cluster render the FCMK more immune to unreliable kernels. They also allow better characterization and adaptation to each individual cluster. Effective kernels tend to contribute more to the clustering and therefore improve results.

3.1 Fuzzy C-means with Multiple Kernels algorithm

To discover nonlinear relationships among data, the proposed Fuzzy C-Means with Multiple Kernels (FCMK) algorithm, like other kernel methods, uses embedding mappings that map features to new feature spaces [105]. We assume that we have a set of M such mappings, $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$. Each mapping ϕ_k encodes the p-dimensional data x

[1]This work has been published in [11]

as a vector $\phi_k(x)$ in its feature space whose dimensionality is L_k. Let $\{K_1, K_2, \ldots, K_M\}$ be the Mercer kernels corresponding to these implicit mappings respectively,

$$K_k(x_j, x_{j'}) = \phi_k(x_j)^T \phi_k(x_{j'}).$$

To combine these kernels and ensure the resulting kernel satisfies Mercer's condition, we consider a non-negative combination of these feature maps within each cluster i, $\phi'^{(i)}$, that is,

$$\phi'^{(i)}(x) = \sum_{k=1}^{M} w_{ik}\phi_k(x) \qquad \text{with } w_{ik} \geq 0,\ \forall i, \text{ and } \sum_{k=1}^{M} w_{ik} = 1.$$

Since these implicit mappings do not necessarily have the same dimensionality, such a linear combination may be impossible. Hence, we construct a new set of independent mappings, $\Psi = \{\psi_1, \psi_2, \ldots, \psi_M\}$, from the original mappings Φ as

$$\psi_1(x) = \begin{bmatrix} \phi_1(x) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \psi_2(x) = \begin{bmatrix} 0 \\ \phi_2(x) \\ \vdots \\ 0 \end{bmatrix}, \ldots, \psi_M(x) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \phi_M(x) \end{bmatrix}.$$

Each of these constructed mappings converts x into an L-dimensional vector, where $L = \sum_{k=1}^{M} L_k$. Constructing new mappings in this way ensures that the feature spaces of these mappings have the same dimensionality and their linear combination can be well defined. In addition, these mappings form a new set of orthogonal bases since

$$\begin{cases} \psi_k(x_j)^T \psi_k(x_{j'}) & = K_k(x_j, x_{j'}) \\ \psi_k(x_j)^T \psi_{k'}(x_{j'}) & = 0 \qquad \text{if } k \neq k' \end{cases}$$

As such orthogonal bases prevent cross terms between different implicit mappings, we can focus on the inner product of data of the same mapping that can be well evaluated by the original kernel functions. We seek to find $\psi^{(i)}(x) = \sum_{k=1}^{M} w_{ik}\psi_k(x)$, a non-negative

linear expansion of the bases in Ψ, which maps data to an implicit feature space with respect to each cluster i. Using the closure properties of kernel functions [105], we construct a new cluster-dependent kernel, $K^{(i)}$, between object j and cluster i. $K^{(i)}$ is computed as a convex combination of M Gaussian kernels $K_1, K_2, \ldots, K_M$ with fixed scaling parameters $\sigma_1, \sigma_2, \ldots, \sigma_M$, respectively. That is,

$$K^{(i)}(x_j, c_i) = \sum_{k=1}^{M} w_{ik}^2 K_k(x_j, c_i). \tag{3.1}$$

In (3.1), $\mathbf{W} = [w_{ik}]$, where $w_{ik} \in [0, 1]$ is a resolution weight for the kernel function K_k with respect to cluster i. A low value of w_{ik} indicates that kernel K_k is not relevant for the density estimation of cluster i (due to its scaling parameter), and that this kernel should not have a significant impact on the creation of this cluster. Similarly, a high value of w_{ik} indicates that the bandwidth of kernel K_k is highly relevant for the density estimation of cluster i, and that this kernel should be the main factor in the creation of this cluster.
We formulate $K^{(i)}$ in terms of Gaussian kernels since they are widely used in literature, they have shown to have strong learning properties in a variety of applications, and they embed the notion of resolution through the scaling parameter σ. Thus, (3.1) becomes

$$K^{(i)}(x_j, c_i) = \sum_{k=1}^{M} \frac{w_{ik}^2}{\sigma_k} \exp\left(-\frac{\|x_j - c_i\|^2}{2\sigma_k^2}\right). \tag{3.2}$$

Note that we add $1/\sigma_k$ to the Gaussian kernel in (3.2) to reduce the effect of extreme cases of σ_k. In fact, as σ_k approaches zero, $K_k(x_j, c_i)$ turns into an impulse function with the value of 1 only at $x_j = c_i$ and 0 elsewhere. In this extreme case, each given data object will no longer have a neighborhood but become a singleton and the distance between any two points in the feature space approaches a common value of 1, leading to difficulties in clustering them. On the other hand, as σ_k approaches infinity, the distance between any two points in the feature space will approach 0 and thus all data will cluster together, leading to a difficulty in separating them.
The Fuzzy Clustering with Multiple Kernels (FCMK) minimizes

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^m \|\psi^{(i)}(x_j) - \psi^{(i)}(c_i)\|^2 \tag{3.3}$$

subject to

$$\sum_{k=1}^{M} w_{ik} = 1, \forall i, \quad \text{and} \quad w_{ik} \geq 0 \ \forall i,k \tag{3.4}$$

$$\sum_{i=1}^{C} \mu_{ij} = 1, \forall j, \quad \text{and} \quad \mu_{ij} \geq 0 \ \forall i,j \tag{3.5}$$

$$\sum_{j=1}^{N} \mu_{ij} > 0, \forall i \tag{3.6}$$

where

$$\psi^{(i)}(x) = \sum_{k=1}^{M} w_{ik} \psi_k(x) \tag{3.7}$$

In (3.3), C and N represent the number of clusters and the number of data points respectively, m is the fuzzification degree usually greater than 1, μ_{ij} is the degree of membership of x_j in cluster i, and c_i is the center of the i^{th} cluster. For brevity, we use D_{ij} to denote the distance between data $\mathbf{x}_j$ and cluster center c_i, i.e.

$$\begin{aligned}
D_{ij}^2 &= \|\psi^{(i)}(x_j) - \psi^{(i)}(c_i)\|^2 \\
&= \left(\psi^{(i)}(x_j) - \psi^{(i)}(c_i)\right)^T \left(\psi^{(i)}(x_j) - \psi^{(i)}(c_i)\right) \\
&= \left(\psi^{(i)}(x_j)^T \psi^{(i)}(x_j)\right) + \left(\psi^{(i)}(c_i)^T \psi^{(i)}(c_i)\right) - 2\left(\psi^{(i)}(x_j)^T \psi^{(i)}(c_i)\right) \\
&= K^{(i)}(x_j, x_j) + K^{(i)}(c_i, c_i) - 2K^{(i)}(x_j, c_i) \\
&= \sum_{k=1}^{M} w_{ik}^2 K_k(x_j, x_j) + \sum_{k=1}^{M} w_{ik}^2 K_k(c_i, c_i) - 2\sum_{k=1}^{M} w_{ik}^2 K_k(x_j, c_i).
\end{aligned} \tag{3.8}$$

The goal of Fuzzy C-Means with Multiple Kernels (FCMK) is to simultaneously find combination weights **W**, memberships **U**, and cluster centers **C** which minimizes the objective function in Equation (3.3).

3.1.1 Optimization of the memberships

Similar to FCM [15], to find the optimal memberships we first fix the weights and cluster centers . Thus, Equation (3.3) can be written as

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^m D_{ij}^2. \tag{3.9}$$

When the weights and cluster centers are fixed, the distances in (3.8) are constant. To optimize (3.9) with respect to the memberships μ_{ij}, we form an energy function with Lagrange multiplier λ for the constraint (3.5) and obtain

$$J_\lambda = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}^m D_{ij}^2 - \sum_{j=1}^{N} \lambda_j \left(\sum_{i=1}^{C} \mu_{ij} - 1 \right). \tag{3.10}$$

By setting the gradient of J_λ to zero, we obtain

$$\frac{\partial J_\lambda}{\partial \lambda} = \sum_{i=1}^{C} \mu_{ij} - 1 = 0, \tag{3.11}$$

and

$$\frac{\partial J_\lambda}{\partial \mu_{ij}} = m\mu_{ij}^{m-1} D_{ij}^2 - \lambda = 0. \tag{3.12}$$

Solving (3.12) for μ_{ij} yields

$$\mu_{ij} = \left(\frac{\lambda}{m} \right)^{\frac{1}{m-1}} \frac{1}{D_{ij}^{2/(m-1)}}. \tag{3.13}$$

Substituting (3.13) back into (3.11), we obtain

$$\sum_{i=1}^{N} \mu_{ij} = \left(\frac{\lambda}{m} \right)^{\frac{1}{m-1}} \sum_{i=1}^{C} \left(\frac{1}{D_{ij}^2} \right)^{\frac{1}{m-1}} = 1. \tag{3.14}$$

Thus,

$$\left(\frac{\lambda}{m} \right)^{\frac{1}{m-1}} = \frac{1}{\sum_{i=1}^{C} \left(\frac{1}{D_{ij}^2} \right)^{\frac{1}{m-1}}}. \tag{3.15}$$

Substituting this expression back in (3.13), we obtain the closed-form solution for the optimal memberships as

$$\mu_{ij} = \frac{1}{\sum_{t=1}^{C} \left(D_{ij}^2 / D_{tj}^2 \right)^{\frac{1}{m-1}}}. \tag{3.16}$$

3.1.2 Optimization of the cluster centers

From Equation (3.16), it can be seen that when the weights **W** and cluster centers **C** are fixed, the optimal memberships **U** can be obtained. Now let us assume the memberships and weights are fixed and derive the optimal centers. Substituting the distance in (3.8) into the FCMK objective function (3.3), we have the following equation

$$J = 2\sum_{i=1}^{C}\sum_{j=1}^{N}\sum_{k=1}^{M}\mu_{ij}^{m}\frac{w_{ik}^{2}}{\sigma_k}\left(1-\exp\left(-\frac{\|x_j-c_i\|^2}{2\sigma_k^2}\right)\right). \tag{3.17}$$

By taking the derivative of J in Equation (3.17) with respect to c_i and setting it to zero, we have

$$\frac{\partial J}{\partial c_i} = \sum_{j=1}^{N}\sum_{k=1}^{M}\mu_{ij}^{m}\frac{w_{ik}^{2}}{\sigma_k^3}(x_j-c_i)\exp\left(-\frac{\|x_j-c_i\|^2}{2\sigma_k^2}\right) = 0.$$

Hence, when **U** are given, the optimal c_i can be computed using

$$c_i = \frac{\sum_{j=1}^{N}\mu_{ij}^{m}\bar{K}^{(i)}(x_j,c_i)x_j}{\sum_{j=1}^{N}\mu_{ij}^{m}\bar{K}^{(i)}(x_j,c_i)}, \tag{3.18}$$

where

$$\overline{K}^{(i)}(x_j,c_i) = \sum_{k=1}^{M}\frac{w_{ik}^{2}}{\sigma_k^3}\exp\left(-\frac{\|x_j-c_i\|^2}{2\sigma_k^2}\right). \tag{3.19}$$

3.1.3 Optimization of the kernel combination weights

Equation (3.8) can be re-arranged as

$$D_{ij}^{2} = \sum_{k=1}^{M}\alpha_{ijk}w_{ik}^{2}, \tag{3.20}$$

where the coefficient α_{ijk} are defined as

$$\alpha_{ijk} = K_k(x_j,x_j) - 2K_k(x_j,c_i) + K_k(c_i,c_i). \tag{3.21}$$

Thus, the objective function in Equation (3.3) becomes

$$J = \sum_{i=1}^{C}\sum_{j=1}^{N}\mu_{ij}^{m}\sum_{k=1}^{M}\alpha_{ijk}w_{ik}^{2}, \tag{3.22}$$

subject to the constraints in (3.4) and (3.5).

When memberships are fixed, (3.22) can be rewritten as

$$J = \sum_{i=1}^{C}\sum_{k=1}^{M}\beta_{ik}w_{ik}^{2}, \tag{3.23}$$

where the coefficient β_{ik} is

$$\beta_{ik} = \sum_{j=1}^{N}\mu_{ij}^{m}\alpha_{ijk}. \tag{3.24}$$

To take the constraint in (3.4) into account, we introduce a Lagrange multiplier, and obtain

$$J_{\lambda} = \sum_{i=1}^{C}\sum_{k=1}^{M}\beta_{ik}w_{ik}^{2} - 2\sum_{i=1}^{C}\lambda_{i}\left(\sum_{k=1}^{M}w_{ik} - 1\right). \tag{3.25}$$

By taking the partial derivatives and setting them to zero, we have

$$\frac{\partial J_{\lambda}}{\partial w_{ik}} = 2\beta_{ik}w_{ik} - 2\lambda = 0. \tag{3.26}$$

The solution of the above equation is $w_{ik} = \frac{\lambda}{\beta_{ik}}$. Because of the constraint $\sum_{k=1}^{M} w_{ik} = 1$, we can eliminate λ and obtain the closed-form solution for the resolution weights

$$w_{ik} = \frac{\frac{1}{\beta_{ik}}}{\sum_{k'=1}^{M}\frac{1}{\beta_{ik'}}}. \tag{3.27}$$

In (3.27), the resolution weight w_{ik} is inversely proportional to α_{ijk}, which is the distance from object x_j to center c_i induced by kernel k. When, the objects are mapped close to the cluster center i, w_{ik} will be large indicating that the kernel k is relevant. On the other hand, when the objects are mapped far away from the center, the weight w_{ik} will be small indicating that the kernel k is irrelevant.

The resulting FCMK approach is outlined in Algorithm 3.1.

Algorithm 3.1 The Fuzzy Clustering with Multiple Kernels (FCMK)

1: Fix the number of clusters C, fuzzification parameter $m \in [1\infty)$;
2: Initialize the fuzzy partition matrix **U**;
3: Initialize the cluster prototypes **C**;
4: Fix the number of kernels M and the scaling parameters $\sigma_1, \ldots, \sigma_M$.
5: **repeat**
6: Compute α_{ijk} coefficients using Equation (3.21);
7: Compute β_{ik} coefficients using Equation (3.24);
8: Update weights using Equation (3.27);
9: Compute the cluster-dependent dissimilarity D_{ij} using Equation (3.8);
10: Update fuzzy memberships using Equation (3.16);
11: Update cluster prototypes using Equation (3.18);
12: **until** (fuzzy memberships do not change or maximum number of iteration is reached).

3.2 Performance illustration

To illustrate the ability of FCMK to learn appropriate local density fitting functions and cluster the data simultaneously, we use it to partition synthetic 2D data sets. We should mention here that, for the purpose of visualizing the results, we use 2-dimensional data. We use 4 data sets that include categories of different shapes with unbalanced sizes and densities. Figure (3.2) displays the 4 synthetic data sets where each cluster is displayed with different color. For the 4 data sets, we set the number of clusters C to the true one (see Figure (3.2)), the fuzzifier m to 1.7, and the maximum number of iterations to 100. The matrix of fuzzy memberships is initialized randomly. The scaling parameters for the kernels are chosen as the prototypes obtained by running the K-Means algorithm on the relational distance with $C = 4$. Therefore, we obtain 4 Gaussian kernels.

As explained in the previous section, FCMK performs density estimation and clustering tasks simultaneously. To assess the robustness of FCMK, we perform extensive experiments on the 4 data sets demonstrating its advantages in term of density estimation and clustering results. We will show the efficacy of the density estimation by evaluating the resolution weights assigned to each kernel induced-similarity. A low value of w_{ik} indicates that the bandwidth σ_k of the kernel K_k is not relevant for the density fitting of cluster i, and that this kernel should not have significant impact on the creation of this cluster. Similarly, a high value of w_{ik} indicates that the bandwidth σ_k of the kernel K_k is highly relevant for fitting the points in cluster i, and that this kernel should be the main factor in the creation of this cluster.

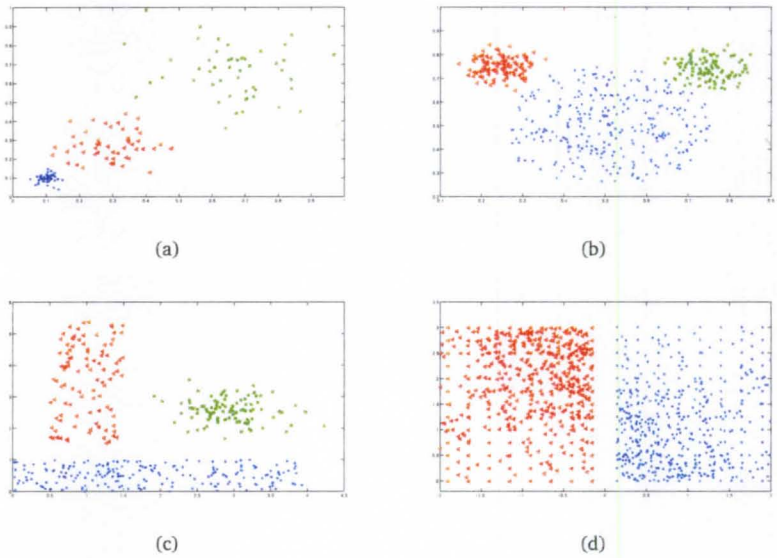
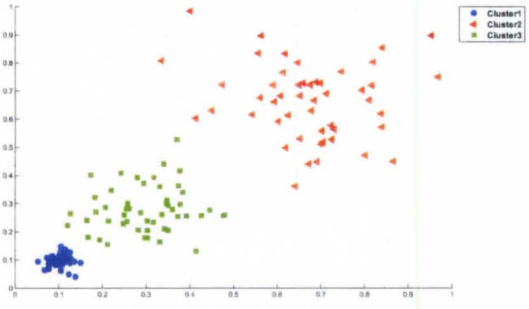
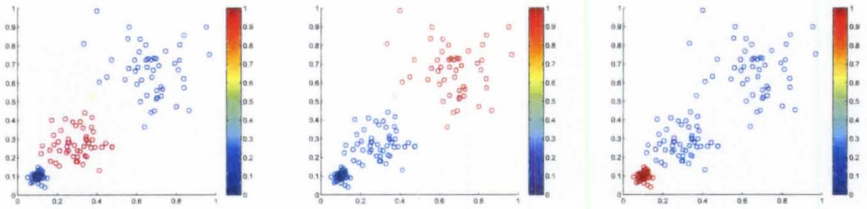


Figure 3.2: Datasets used to illustrate the performance of FCMK. Each cluster is shown by a different color.

Table 3.1: Partition and resolution weights learned by FCMK for dataset 1 displayed in Figure 3.2(a)



	$\sigma_k = 0.01$	$\sigma_k = 0.08$	$\sigma_k = 0.24$	$\sigma_k = 0.45$	True (σ_x, σ_y)
Cluster 1	0	0.146	0.534	0.320	$\sigma_{cl1} = (0.14; 0.13)$
Cluster 2	0.398	0.547	0.055	0	$\sigma_{cl2} = (0.09; 0.07)$
Cluster 3	0.705	0.295	0	0	$\sigma_{cl3} = (0.02; 0.02)$



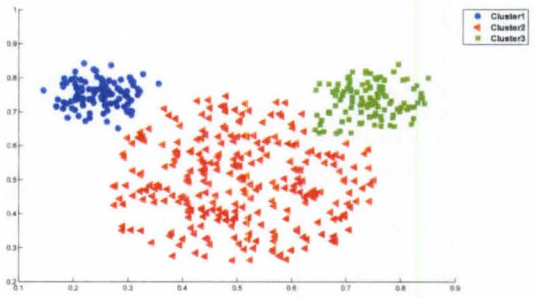
(a) (b) (c)

Figure 3.3: Fuzzy memberships learned by FCMK on dataset 1 (Figure 3.2(a)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

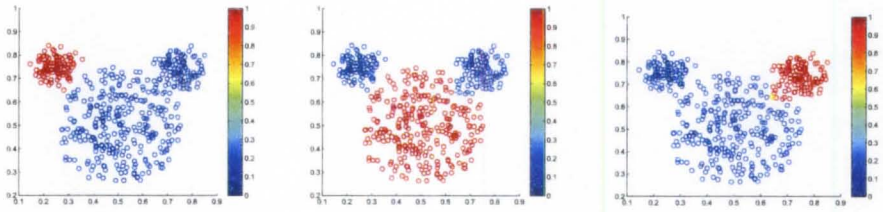
Table (3.1) displays the clustering results and the resolution weights learned by FCMK on dataset 1. Dataset 1 includes three Gaussian clusters that have the same size but varying densities. First, we notice that the estimated cluster-dependent resolution weights reflect the geometry of the data. More specifically, they are related to the variance of the clusters. In Table (3.1), FCMK assigns higher weights to $\sigma_k = 0.24$ and $\sigma_k = 0.45$ for cluster 1. In fact, the true standard deviation of cluster 1 is $\sigma_{cl1} = (0.14, 0.13)$ which explains the importance given to $\sigma_k = 0.24$ and $\sigma_k = 0.45$ and the zero weight given to the irrelevant resolution $\sigma_k = 0.01$. Cluster 3 has smaller spatial variance than cluster 1. Therefore, the higher weights were assigned to $\sigma_k = 0.01$ and $\sigma_k = 0.08$ and zero weight for $\sigma_k = 0.24$ and $\sigma_k = 0.45$. By learning appropriate resolution weights, FCMK partitions this data correctly. Figure (3.3) displays the fuzzy memberships with respect to each cluster. We can notice that most membership values tend to be binary (either 0 or 1). This means that the learned multiple kernels have mapped the data to well separated clusters in the feature space.

The partitioning of dataset 2 is reported in Table (3.2). The geometry of this data is more complex. The clusters are close to each others and have different densities and sizes. The kernel weights learned by FCMK reflect the distribution of each cluster. For instance, cluster 2 has equal weights for the different values of σ_k. This reflects the fact that points within this cluster are dispersed and some of them are closer to cluster 1 and 3 than other points in this cluster. These learned weights ensure that cluster 2 does not include points from other clusters. On the other hand, FCMK assigns relatively higher weight to $\sigma_k = 0.01$ for the two small and dense clusters as reported in Table (3.2). We observe that some boundary points are not categorized correctly. In fact, as it can be seen in Figure (3.4), the fuzzy membership of these points is around 0.5. This is due to

Table 3.2: Partition and resolution weights learned by FCMK for dataset 2 displayed in Figure 3.2(b)



	$\sigma_k = 0.01$	$\sigma_k = 0.05$	$\sigma_k = 0.1$	$\sigma_k = 0.5$	True (σ_x, σ_y)
Cluster 1	0.275	0.337	0.203	0.185	$\sigma_{cl1} = (0.04; 0.04)$
Cluster 2	0.243	0.247	0.259	0.251	$\sigma_{cl2} = (0.12; 0.12)$
Cluster 3	0.275	0.338	0.203	0.184	$\sigma_{cl3} = (0.04; 0.04)$



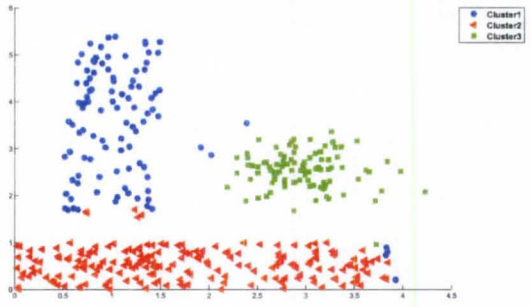
(a) (b) (c)

Figure 3.4: Fuzzy memberships learned by FCMK on dataset 2 (Figure 3.2(b)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

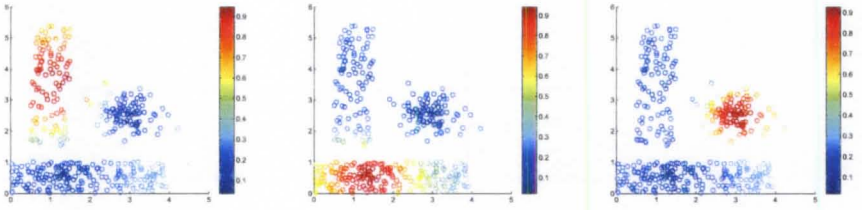
the characteristic of this dataset where the boundaries are not well defined.

A similar analysis on the learned resolution weights for dataset 3 is conducted and reported in Table (3.3). The learned resolution weights found by FCMK makes it possible to identify the 3 clusters correctly. In Table (3.3), cluster 2 has a rectangular shape with a true standard deviation of 1.14 in the x-axis and 0.28 in the y-axis. The geometry of cluster 2 is captured by our algorithm. In fact, higher weights were assigned to $\sigma_k = 0.55$ and $\sigma_k = 0.85$ which reflect the high variance according to the x-axis and the low variance according to the y-axis, respectively. This shows that FCMK can find resolution weights effectively to fit each cluster and obtain the correct categorization of the data. Figure (3.5) shows that some points have fuzzy memberships around 0.5, indicating that they are close to more than one cluster in the feature space. This is an inherit limitation

Table 3.3: Partition and resolution weights learned by FCMK for dataset 3 displayed in Figure 3.2(c)



	$\sigma_k = 0.16$	$\sigma_2 = 0.55$	$\sigma_3 = 0.85$	$\sigma_4 = 1.4$	True (σ_x, σ_y)
Cluster 1	0.062	0.363	0.387	0.188	$\sigma_{cl1} = (0.29; 1.17)$
Cluster 2	0.064	0.363	0.386	0.187	$\sigma_{cl2} = (1.14; 0.28)$
Cluster 3	0.335	0.448	0.217	0	$\sigma_{cl3} = (0.41; 0.37)$



(a) (b) (c)

Figure 3.5: Fuzzy memberships learned by FCMK on dataset 3 (Figure 3.2(c)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

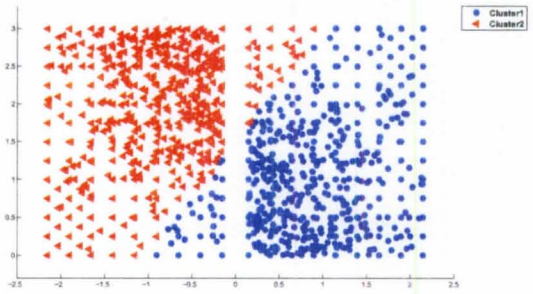
of FCMK since it implicitly uses the Euclidean distance to map the data.

The clustering results returned by FCMK on dataset 4 (Table 3.4) are not meaningful. In fact, dataset 4 is constituted of two clusters close to each others with multiple resolutions within each cluster. This does not correspond to the standard way of perceiving the intra-cluster and the inter-cluster dissimilarities. That is why the resolution weights learned by FCMK fail to deal with the geometric characteristics of this dataset.

3.3 Limitations of FCMK

The proposed FCMK can handle all these cases and can perform particularly well on data sets 1 and 2 which cannot be handled satisfactory by other methods. These two data sets

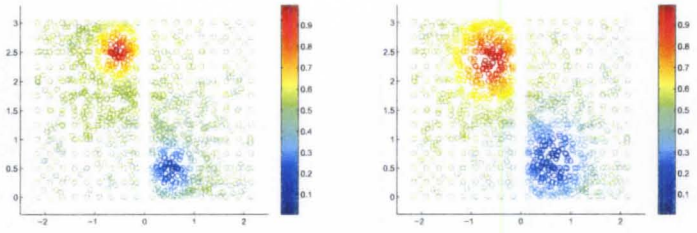
Table 3.4: Partition and resolution weights learned by FCMK for dataset 4 displayed in Figure 3.2(d)



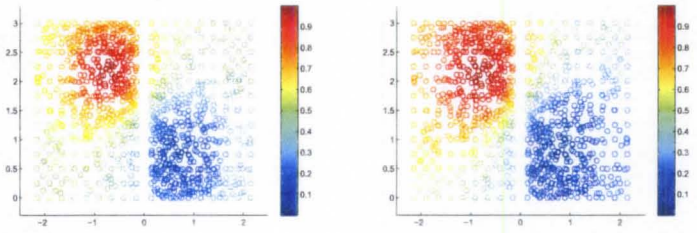
	$\sigma_k = 0.1$	$\sigma_k = 0.5$	$\sigma_k = 1$	$\sigma_k = 1.5$	True (σ_x, σ_y)
Cluster 1	0.248	0.254	0.253	0.245	$\sigma_{cl1} = (0.56; 0.81)$
Cluster 2	0.244	0.258	0.255	0.243	$\sigma_{cl2} = (0.56; 0.81)$

are much more difficult cases as the clusters are close to each others. Moreover, there are large variations between the distributions and the densities of each cluster. As it can be seen in Table 3.1 and Table 3.2, FCMK have succeeded to learn resolution weights with respect to each cluster. These weights estimate the density of each cluster and thus lead our proposed algorithm to a satisfactory categorization of the data. The experimental results presented in this chapter demonstrate the effectiveness of the proposed method both in terms of categorization performance and in terms of density estimation. The FCMK algorithm is simple, quite efficient and can be used for a wide variety of data types.

Since it is based on KFCM [124], FCMK inherits the limitations of object-based clustering. In particular, multiple runs of the algorithm may be needed since FCMK is susceptible to initialization problems. Moreover, FCMK is not suitable for all types of data. In FCMK, the density is estimated by counting the number of points within a specified radius, σ_k, of the center as seen in Figure (3.6). However, we seek to estimate the variance between pairs of points and not point to center to detect the multiple resolutions within the same cluster. Besides, FCMK is not robust to noise and outliers. Outlier detection and removal can help significantly in such situations. Finally, FCMK is restricted to data for which there is a notion of center (centroid). In some applications, the set of objects may be represented by relational data.



(a) $\sigma = 0.1$ (b) $\sigma = 0.5$



(c) $\sigma = 1$ (d) $\sigma = 1.5$

Figure 3.6: Density estimation for dataset 4 (Figure 3.2(d))

In order to address these drawbacks, in the next chapter, we propose a new approach called Relational Fuzzy Clustering with Multiple Kernels (RFCMK) to cluster relational data.

CHAPTER 4

RELATIONAL DATA CLUSTERING WITH MULTIPLE KERNELS

Recently, relational clustering has become an active field of research due to its ability to use the adjacency structure of the data and avoid dealing with a prefixed shape of clusters. In this chapter, we propose a new relational fuzzy clustering with multiple kernels algorithm (RFCMK)[1] to overcome the limitations of FCMK described in Section 3.3. For instance, the RFCMK is general in the sense that it is applicable when data cannot be represented by feature vectors and only the degree to which pairs of objects in the data are related is available. Moreover, even if the data can be represented by feature vectors, the RFCMK is more practical when similar objects cannot be represented efficiently by a single prototype. RFCMK strives to find a good partitioning of the data into meaningful clusters, together with the optimal kernel-induced feature map in a completely unsupervised way.

The RFCMK algorithm is a generalization of the Kernelized Non-Euclidean Relational Fuzzy (kNERF) C-Means algorithm [88] and uses a new optimization criterion to estimate the optimal convex combination of multiple kernels with respect to each cluster in a completely unsupervised manner. Similar to the FCMK algorithm, it constructs the kernel from a number of multi-resolution Gaussian kernels and learns a resolution weight for each kernel function in each cluster. This allows better characterization and adaptivity to each individual cluster.

[1] This work has been published in [12]

The RFCMK algorithm is capable of discovering multi-resolution clusters with different shapes and non-linear boundaries in the input space. It also benefits from the non-linear capabilities of kernel methods without the hassle of kernel parameters tuning.

4.1 Relational Fuzzy C-means with Multiple Kernels algorithm

In the following, we assume that $\{x_1, \ldots, x_N\}$ is a set of N data points to be partitioned into C clusters. Let $\mathbf{R} = [r_{jh}]$ be a relational matrix where r_{jh} represents the degree to which pairs of objects x_j and x_h are related. The matrix $\mathbf{R}$ could be given or constructed from the features of the objects. Each object x_j belongs to cluster i with fuzzy membership μ_{ij} that satisfies the constraints in (3.5) and (3.6).

Similarly to FCMK, we construct a new similarity, $K^{(i)}$, with respect to cluster i between objects x_j and x_h. $K^{(i)}$ is computed as a linear combination of M Gaussian kernels $K_1, \ldots, K_M$, with fixed scaling parameters $\sigma_1, \ldots, \sigma_M$, respectively. That is, each x_j is translated via M mappings $\psi_k(x_j) \mapsto \mathbb{R}^L$, $k = 1, \ldots, M$ from the input space into M higher dimensional feature spaces $\left(\psi_1(x_j), \ldots, \psi_M(x_j)\right)$ where $L = \sum_{k=1}^{M} L_k$ denotes the dimensionality of the k^{th} feature space. Formally the kernel $K^{(i)}$ is defined as

$$\begin{aligned} K^{(i)}(x_j, x_h) &= \sum_{k=1}^{M} w_{ik}^2 K_k(x_j, x_h) \\ &= \sum_{k=1}^{M} w_{ik}^2 \exp\left(\frac{\|x_j - x_h\|^2}{2\sigma_k^2}\right). \end{aligned} \tag{4.1}$$

In (4.1), $\mathbf{W} = [w_{ik}]$, where $w_{ik} \in [0, 1]$ is a resolution weight for the kernel matrix K_k with respect to cluster i that satisfies the constraint in (3.4). A low value of w_{ik} indicates that the bandwidth of the kernel K_k is not relevant for the density estimation of cluster i, and that this matrix should not have a significant impact on the creation of this cluster. Similarly, a high value of w_{ik} indicates that the bandwidth of kernel K_k is highly relevant for the density estimation of cluster i, and that this matrix should be the main factor in the creation of this cluster.

The Relational Fuzzy Clustering with Multiple Kernels algorithm (RFCMK) minimizes

$$J = \sum_{i=1}^{C} \frac{\sum_{j=1}^{N} \sum_{h=1}^{N} \mu_{ij}^{m} \mu_{ih}^{m} \widehat{r}_{jh}^{(i)}}{\sum_{h=1}^{N} \mu_{ih}^{m}}, \tag{4.2}$$

subject to

$$\mu_{ij} \in [0,1] \quad \text{and} \quad \sum_{i=1}^{C} \mu_{ij} = 1 \ \forall j = 1, \ldots, N, \tag{4.3}$$

and

$$w_{ik} \in [0,1] \quad \text{and} \quad \sum_{k=1}^{M} w_{ik} = 1 \ \forall i = 1, \ldots, C. \tag{4.4}$$

In (4.2), C is the number of clusters, $m \in [1, \infty)$ is the fuzzifier, μ_{ij} is the fuzzy membership of object x_j in cluster i, and $\widehat{r}_{jh}^{(i)}$ is the transformed relational data between feature points x_j and x_h, with respect to cluster i. Using the implicit mapping $\psi^{(i)}$, the transformed dissimilarity $\widehat{r}_{jh}^{(i)}$ can be defined as

$$\begin{aligned}
\widehat{r}_{jh}^{(i)} &= \|\psi^{(i)}(x_j) - \psi^{(i)}(x_h)\|^2 \\
&= \psi^{(i)}(x_j)^T \psi^{(i)}(x_j) + \psi^{(i)}(x_h)^T \psi^{(i)}(x_h) - 2\psi^{(i)}(x_j)^T \psi^{(i)}(x_h) \\
&= K^{(i)}(x_j, x_j) + K^{(i)}(x_h, x_h) - 2K^{(i)}(x_j, x_h) \\
&= \sum_{k=1}^{M} w_{ik}^2 K_k(x_j, x_j) + \sum_{k=1}^{M} w_{ik}^2 K_k(x_h, x_h) - 2 \sum_{k=1}^{M} w_{ik}^2 K_k(x_j, x_h).
\end{aligned} \tag{4.5}$$

We should note here that the objective function in (4.2) includes only the transformed dissimilarities $\widehat{r}_{jh}^{(i)}$ and the membership values that we are trying to learn. Moreover, it involves dissimilarity between pairs of objects (instead of dissimilarity of the objects to a cluster center). Thus, object data is not required for the RFCMK method. The goal of the clustering algorithm is to identify the resolution weight w_{ik} and the membership values μ_{ij} by solving an optimization problem. In order to compute the optimal values of w_{ik} and μ_{ij}, we use the well known alternating optimization method.

4.1.1 Optimization of the memberships

In order to formulate the relational dual of the Fuzzy C-means algorithm, Hathaway et al. [84] proved that the squared Euclidean distance $D_{ij}^2 = \|x_j - c_i\|^2$, from feature vector x_j to the center of the i^{th} cluster, c_i, can be written in terms of the relational matrix $\mathbf{R}$ as follows

$$D_{ij}^2 = \left(Rv_i\right)_j - \frac{v_i^T R v_i}{2}, \tag{4.6}$$

where v_i is the membership vector defined by

$$v_i = \frac{\left(\mu_{i1}^m, \ldots, \mu_{iN}^m\right)^T}{\sum_{j=1}^N \mu_{ij}^m}. \tag{4.7}$$

Equation (4.6) allows the computation of the distance between the data points and cluster prototypes in each iteration when only relational data, $\mathbf{R}$, are given, starting with a set of initial fuzzy memberships, μ_{ij}, that describe the degree of belongingness of the j^{th} data object to the i^{th} cluster. Once the implicit distance values, D_{ij}, have been computed using (4.6), the fuzzy memberships can be updated to optimize the clustering criterion, resulting in a new fuzzy partition of the data.

Substituting the dissimilarity $\mathbf{R}$ in the dual relation (4.6) by the new defined dissimilarity $\widehat{R}^{(i)} = [\widehat{r}_{jh}^{(i)}]$ gives

$$D_{ij}^2 = \left(\widehat{R}^{(i)} v_i\right)_j - \frac{v_i^T \widehat{R}^{(i)} v_i}{2}, \tag{4.8}$$

Using the implicit distance values, D_{ij}, the objective function in (4.2) could be written as:

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^m D_{ij}^2 \tag{4.9}$$

To optimize J with respect to μ_{ij} subject to (4.3), we use the Lagrange multiplier technique and obtain

$$J_\lambda = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^m D_{ij}^2 - \sum_{j=1}^{N} \lambda_j \left(\sum_{i=1}^{C} \mu_{ij} - 1 \right). \tag{4.10}$$

By setting the gradient of J_λ to zero, we obtain

$$\frac{\partial J_\lambda}{\partial \lambda} = \sum_{i=1}^{C} \mu_{ij} - 1 = 0, \tag{4.11}$$

and

$$\frac{\partial J_\lambda}{\partial \mu_{ij}} = m\mu_{ij}^{m-1} D_{ij}^2 - \lambda = 0. \tag{4.12}$$

Equation (4.12) yields

$$\mu_{ij} = \left(\frac{\lambda}{m D_{ij}^2} \right)^{\frac{1}{m-1}}. \tag{4.13}$$

Substituting (4.13) back into (4.11), we obtain

$$\sum_{i=1}^{C} \mu_{ij} = \left(\frac{\lambda}{m} \right)^{\frac{1}{m-1}} \sum_{i=1}^{C} \left(\frac{1}{D_{ij}^2} \right)^{\frac{1}{m-1}} = 1. \tag{4.14}$$

Thus,

$$\left(\frac{\lambda}{m} \right)^{\frac{1}{m-1}} = \frac{1}{\sum_{i=1}^{C} \left(\frac{1}{D_{ij}^2} \right)^{\frac{1}{m-1}}}. \tag{4.15}$$

Substituting this expression back in (4.13), we obtain the following update equation

$$\mu_{ij} = \frac{1}{\sum_{t=1}^{C} \left(\frac{D_{ij}^2}{D_{tj}^2} \right)^{\frac{1}{m-1}}}. \tag{4.16}$$

Some of the distances computed using (4.8) may be negative. To overcome this, we adopt the solution used in the Non-Euclidean Relational Fuzzy (NERF) C-Means algorithm [52]. NERF C-Means modifies the RFCM by adding a step that converts the non-Euclidean distance $\widehat{R}^{(i)}$ into Euclidean distance $\widehat{R}_\beta^{(i)}$ as follows

$$\widehat{R}_\beta^{(i)} = \widehat{R}^{(i)} + \beta \mathbf{x}(M - I). \tag{4.17}$$

In (4.17), β is a suitably chosen scalar, $I \in \mathbb{R}^{N \mathbf{x} N}$ is the identity matrix, and $M \in \mathbb{R}^{N \mathbf{x} N}$ has all its entries equal to one. The lower bound by which it is necessary to shift β to make the distances positive is derived in [52] to be

$$\Delta\beta = \max_{ih} \left\{ -2D_{ih}^2 / \|v_i - e_h\|^2 \right\} \tag{4.18}$$

where e_h denotes the h^{th} column of the identity matrix.

4.1.2 Optimization of the kernel combination weights

Equation (4.5) can be re-arranged as

$$\widehat{r}_{jh}^{(i)} = \sum_{k=1}^{M} \alpha_{jhk} w_{ik}^2, \tag{4.19}$$

where the coefficient α_{jhk} can be written as

$$\alpha_{jhk} = K_k(x_j, x_j) + K_k(x_h, x_h) - 2K_k(x_j, x_h). \tag{4.20}$$

Substituting (4.19) back in (4.2), the objective function of RFCMK becomes

$$J = \sum_{i=1}^{C} \frac{\sum_{j=1}^{N} \sum_{h=1}^{N} \mu_{ij}^m \mu_{ih}^m \sum_{k=1}^{M} w_{ik}^2 \alpha_{jhk}}{2 \sum_{h=1}^{N} \mu_{ih}^m} \tag{4.21}$$

To optimize J with respect to the kernel weights W, we fix the memberships and use the Lagrange multiplier technique. We obtain

$$J_\lambda = \sum_{i=1}^{C} \frac{\sum_{j=1}^{N} \sum_{k=1}^{N} \mu_{ij}^m \mu_{ik}^m \sum_{l=1}^{M} w_{il}^2 \alpha_{jkl}}{2 \sum_{k=1}^{N} \mu_{ik}^m} - \sum_{i=1}^{C} \lambda_i \left(\sum_{l=1}^{M} w_{il} - 1 \right). \tag{4.22}$$

Since the rows of W are independent of each other, we can reduce the above optimization problem to the following C independent problems:

$$J_{\lambda_i} = \frac{\sum_{j=1}^{N} \sum_{h=1}^{N} \mu_{ij}^m \mu_{ih}^m \sum_{k=1}^{M} w_{ik}^2 \alpha_{jhk}}{2 \sum_{h=1}^{N} \mu_{ih}^m} - \lambda_i \left(\sum_{k=1}^{M} w_{ik} - 1 \right), \text{ for } i = 1, \ldots, C.$$

By setting the gradient of J_{λ_i} to zero, we obtain

$$\frac{\partial J_{\lambda_i}}{\partial \lambda_i} = \left(\sum_{k=1}^{M} w_{ik} - 1 \right) = 0 \tag{4.23}$$

and

$$\frac{\partial J_{\lambda_i}}{\partial \lambda_i} = 2w_{ik} \frac{\sum_{j=1}^{N} \sum_{h=1}^{N} \mu_{ij}^m \mu_{ih}^m \alpha_{jhk}}{2 \sum_{h=1}^{N} \mu_{ih}^m} - \lambda_i = 0 \tag{4.24}$$

Solving (4.23) and (4.24) for w_{ik}, we obtain

$$w_{ik} = \frac{1}{\sum_{p=1}^{M} \left(\bar{D}_{ik} / \bar{D}_{ip} \right)}, \tag{4.25}$$

where

$$\bar{D}_{ik} = \sum_{j=1}^{N} \sum_{h=1}^{N} \mu_{ij}^m \mu_{ih}^m \alpha_{jhk}. \tag{4.26}$$

In (4.25), the resolution weight w_{ik} is inversely proportional to α_{jhk}, which is the distance between objects x_j and x_h induced by kernel k. When, the objects are mapped close to each other, w_{ik} will be large indicating that the kernel k is relevant. On the other hand, when, the objects are mapped far apart, the weight w_{ik} will be small indicating that the kernel k is irrelevant.

The RFCMK algorithm is summarized in Algorithm 4.1.

Algorithm 4.1 The RFCMK Algorithm

Fix the number of clusters C, $m \in [1, \infty)$, the number of kernels M, and kernel parameters $\sigma_1, \ldots, \sigma_M$;
Initialize the fuzzy partition matrix U;
Initialize $\beta = 0$, and $w_{ik} = 1/M$;
repeat
  Compute total dissimilarities $\widehat{R}^{(i)}$ using (4.5);
  Compute $\widehat{R}_{\beta}^{(i)}$ using (4.17);
  Compute membership vectors v_i using (4.7);
  Compute distances using (4.8);
  if $d_{ik}^2 < 0$ for any i, k **then**
    Compute $\Delta\beta$ using (2.19);
    $D_{ih}^2 = D_{ih}^2 + (\Delta\beta/2) * \|v_i - e_k\|$;
    $\beta = \beta + \Delta\beta$;
  end if
  Update fuzzy memberships using (4.16);
  Update kernel combination weights using (4.25);
until (fuzzy memberships do not change).

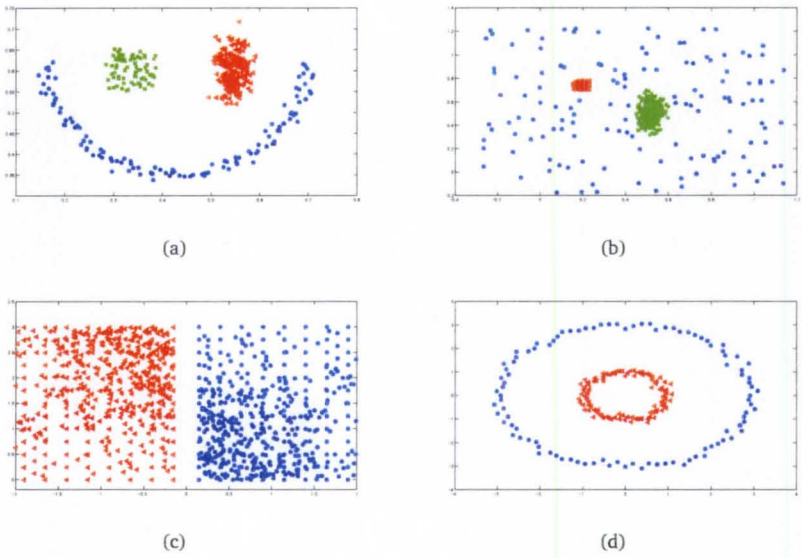


Figure 4.1: Datasets used to illustrate the performance of RFCMK. Each cluster is shown by a different color.

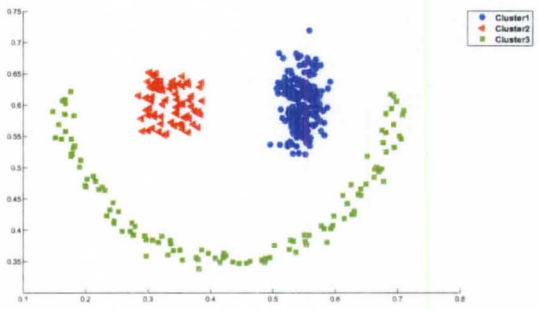
4.2 Performance illustration

To illustrate the ability of the RFCMK algorithm to learn appropriate local density fitting functions and cluster the data simultaneously, we use it to categorize synthetic 2D data sets. We use four data sets that include categories with unbalanced sizes and densities. Figure (4.1) displays the four synthetic data sets where each cluster is displayed by a different color. For the four data sets, we set the number of clusters C to the true one (see Figure (4.1)), the fuzzifier m to 1.7, and the maximum number of iterations to 100. The matrix of fuzzy memberships is initialized randomly. The scaling parameters for the kernels are chosen in the same way as the FCMK. That is, they are set to the prototypes obtained by running the K-Means algorithm on all pairwise distances with $C = 4$.

For each data set, we validate the usefulness of the proposed algorithm by analyzing the resolution weights assigned to each kernel induced similarity. A low value of w_{ik} indicates that the bandwidth σ_k of the kernel K_k is not relevant for the density fitting of cluster i, and that this matrix should not have significant impact on the creation of this cluster. Similarly, a high value of w_{ik} indicates that the bandwidth σ_k of the kernel K_k is highly relevant for fitting the points in cluster i, and that this matrix should be the main factor in the creation of this cluster.

Table (4.1) displays the partition and the resolution weights learned by RFCMK on

Table 4.1: Partition and resolution weights learned by RFCMK for dataset 5 displayed in Figure (4.1(a))

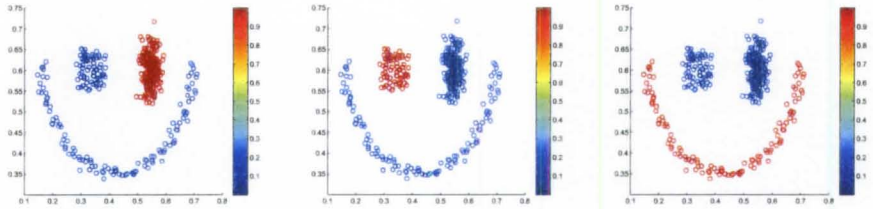


	$\sigma_k = 0.02$	$\sigma_k = 0.04$	$\sigma_k = 0.06$	$\sigma_k = 0.1$	True σ
Cluster 1	0.030	0.417	0.309	0.244	$\sigma_{cl1} = 0.03$
Cluster 2	0.027	0.429	0.309	0.235	$\sigma_{cl2} = 0.03$
Cluster 3	0.031	0.188	0.288	0.493	$\sigma_{cl3} = 0.15$

dataset 5. The RFCMK algorithm is able to partition this data correctly by learning resolution weights with respect to each cluster. As it can be seen in Table 4.1, RFCMK assigns higher weights to $\sigma_k = 0.04$ and $\sigma_k = 0.06$ for clusters 1 and 2. In fact, the pairwise standard deviation of clusters 1 and 2 is $\sigma = 0.03$ which explains the importance given to $\sigma_k = 0.04$ and $\sigma_k = 0.06$. Cluster 3 has a standard deviation of 0.15. Thus, RFCMK assigns the highest weight to $\sigma_k = 0.1$. When we learn resolution weights with respect to each cluster as in RFCMK, the dissimilarity metric is more adaptable to the geometry of the data set.

The fuzzy memberships of all points in the three clusters are displayed in Figure (4.2). As it can be seen, most of the points belong to one of the clusters with a membership larger than 0.9. This indicates that the mapping using the learned w_{ik}'s makes the three clusters well separated in the feature space.

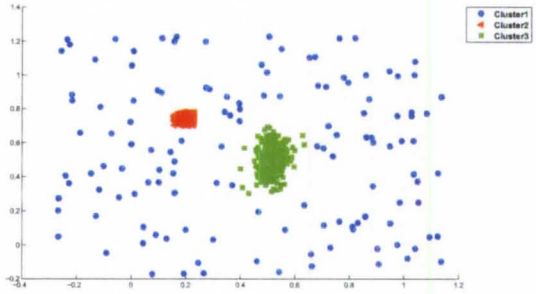
Dataset 6 displayed in Figure (4.1(b)) is more complex and the clusters are close to each others with unbalanced sizes and densities. The resolution weights learned by RFCMK make it possible to identify the three clusters correctly (refer to Table 4.2). For instance, RFCMK assigns higher weights to $\sigma_k = 0.02$ and $\sigma_k = 0.05$ for cluster 2 whose true pairwise standard deviation is equal to 0.02. For cluster 1 with a standard deviation of 0.35, the highest resolution weight is assigned to $\sigma_k = 0.5$. This illustrates the ability



(a) (b) (c)

Figure 4.2: Fuzzy memberships learned by RFCMK on dataset 5 (Figure (4.1(a))) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

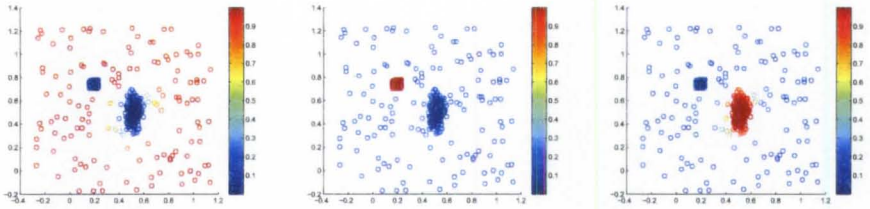
Table 4.2: Partition and resolution weights learned by RFCMK for dataset 6 displayed in Figure (4.1(b))



	$\sigma_k = 0.02$	$\sigma_k = 0.05$	$\sigma_k = 0.1$	$\sigma_k = 0.5$	True σ
Cluster 1	0.032	0.092	0.135	0.741	$\sigma_{cl1} = 0.35$
Cluster 2	0.520	0.361	0.092	0.027	$\sigma_{cl2} = 0.02$
Cluster 3	0.058	0.235	0.538	0.169	$\sigma_{cl3} = 0.06$

of RFCMK to learn good resolution weights according to the spatial variance of the data set. We also notice from Figure (4.3) that the points lying at the boundaries separating the three clusters have fuzzy memberships in the 0.5 range, thus reflecting the geometric characteristic of this dataset.

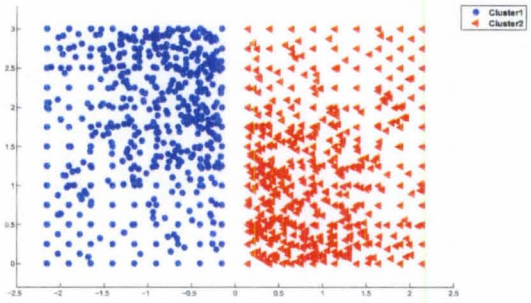
Table 4.3 reports the results obtained by partitioning dataset 4 using RFCMK. We should recall that FCMK was not able to partition this data correctly (refer to Table 3.4). On the other hand, the RFCMK algorithms can distinguish the different intrinsic sub-structures of clusters. Consequently, it learns different kernel combinations to capture the multi-resolution within each cluster. Figure (4.4) displays the fuzzy memberships with respect to each cluster. We can notice that most membership values tend to be binary (either 0 or 1). This means that the learned multiple kernels have mapped the data to well separated



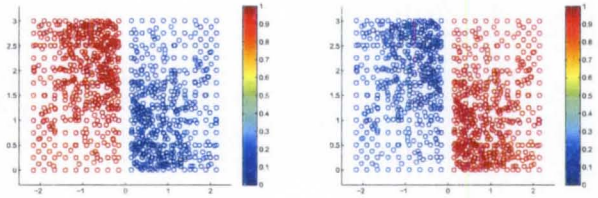
(a) (b) (c)

Figure 4.3: Fuzzy memberships learned by RFCMK on dataset 6 (Figure (4.1(b))) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

Table 4.3: Partition and resolution weights learned by RFCMK for dataset 4 displayed in Figure (3.2(d))



	$\sigma_k = 0.1$	$\sigma_k = 0.5$	$\sigma_k = 1$	$\sigma_k = 1.5$	True σ
Cluster 1	0.157	0.292	0.291	0.260	$\sigma_{cl1} = 0.68$
Cluster 2	0.153	0.295	0.298	0.254	$\sigma_{cl2} = 0.68$

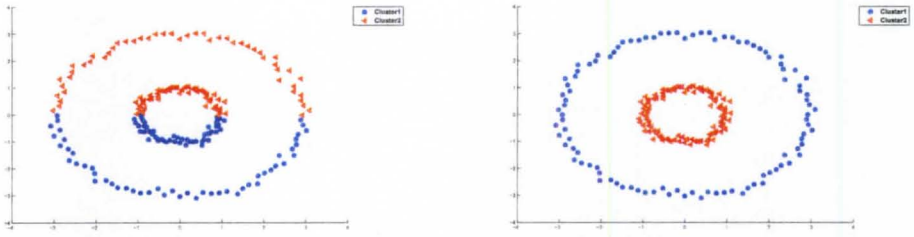


(a) (b)

Figure 4.4: Fuzzy memberships learned by RFCMK on dataset 4 (Figure (3.2(d))) with respect to (a) cluster 1, and (b) cluster 2.

clusters in feature space.

The clustering results returned by FCMK on dataset 7 (refer to Figure (4.5(a))) are not meaningful. On the other hand, RFCMK succeeds to partition the two co-centric rings as shown in Figure (4.5(b)). This is due to the fact that RFCMK allows to find a soft linear



(a) (b)

Figure 4.5: Partitions of dataset 7 displayed in Figure (4.1(d)) learned by (a) FCMK and (b) RFCMK

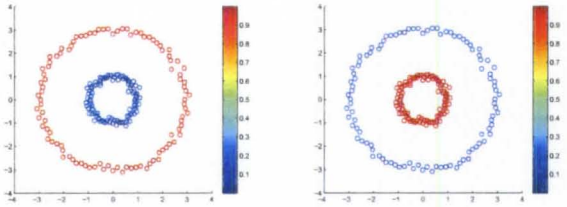
Table 4.4: Resolution weights learned by RFCMK for dataset 7 displayed in Figure (4.1(d))

	$\sigma_k = 0.25$	$\sigma_k = 0.70$	$\sigma_k = 1$	$\sigma_k = 1.5$	True σ
Cluster 1	0.234	0.245	0.254	0.267	$\sigma_{cl1} = 1.21$
Cluster 2	0.183	0.344	0.258	0.215	$\sigma_{cl2} = 0.71$

partitioning of the feature space. This partitioning, back to the input space, results in a soft non-linear partitioning of data. Thus, this method is more suitable for non-linear data clustering than FCMK. Moreover, FCMK is a prototype-based clustering approach. However, groups of similar objects in dataset 7 cannot be represented efficiently by a single prototype.

Table 4.4 reports the learned cluster weight in each kernel induced Hilbert space. It can be seen that the results are consistent with the intuition. For example, $\sigma_k = 0.7$ dominates cluster 2, and for cluster 1, kernels with larger widths gradually increase their relative weights. Figure (4.6) shows that most of the points belong to one of the clusters with a high membership. This is due to the implicit mapping of the data into a high dimensional space using the learned resolution weights. Computing a linear partitioning in this feature space results in a nonlinear partitioning in the input space.

In this chapter, we presented a relational approach that learns local combination of Gaussian kernels for each cluster and partition the data simultaneously. We showed that the RFCMK algorithm gives satisfactory clustering results on 2D datasets. Moreover, we showed that the learned resolution weights and the fuzzy memberships returned by RFCMK are meaningful and reflect the geometric characteristic of the data.



(a) (b)

Figure 4.6: Fuzzy memberships learned by RFCMK on dataset 7 (Figure (4.1(d))) with respect to (a) cluster 1, and (b) cluster 2.

The FCMK and RFCMK objective functions have several parameters and their optimization is prone to several local minima and is sensitive to initialization. This problem is more acute for high dimensional dataset. Thus, if a small amount of prior knowledge is available, it can be used to guide the clustering algorithms to avoid most local minima and obtain a better partition. In the next chapter, we present semi-supervised versions of these algorithms that use a small amount of side information.

CHAPTER 5

SEMI-SUPERVISED CLUSTERING WITH MULTIPLE KERNELS

Clustering is a difficult combinatorial problem that is susceptible to local minima, especially for high dimensional real world data. Incorporating prior knowledge in the unsupervised learning task, in order to guide the clustering process has attracted considerable interest among researchers in the data mining and machine learning communities. This prior knowledge is usually available in the form of hints, constraints, or labels. Supervision in the form of constraints is more practical than providing class labels. This is because in many real world applications, the true class labels may not be known, and it is much easier to specify whether pairs of points should belong to the same or to different clusters. In fact, pairwise constraints occur naturally in many domains.

In this chapter, we present two semi-supervised clustering approaches: the Semi-Supervised Fuzzy Clustering with Multiple Kernels (SS-FCMK), and the Semi-Supervised Relational Fuzzy Clustering with Multiple Kernels (SS-RFCMK). SS-FCMK and SS-RFCMK are search-based algorithms that allow the constraints to guide the clustering process towards an appropriate partition.

Let $\boldsymbol{ML}$ be the set of "*Must-Link*" constraints, i.e. $(x_j, x_h) \in \boldsymbol{ML}$ implies that x_j and x_h should be assigned to the same cluster. Similarly, let $\boldsymbol{CL}$ the set of "*Cannot-Link*" constraints, i.e.$(x_j, x_h) \in \boldsymbol{CL}$ implies that x_j and x_h should be assigned to different clusters. In standard semi-supervised clustering [55, 36], the pairwise constraints are crisp, treated equally important, and enforced during optimization. In our formulation, the pairwise

constraints used in SS-FCMK and SS-RFCMK are soft reflecting the uncertainty associated with *a priori* knowledge about the pair of points that should or should not belong to the same cluster. Thus, they are viewed as recommendations.

5.1 The semi-supervised prototype-based clustering with multiple kernels

5.1.1 Semi-Supervised Fuzzy Clustering with Multiple Kernels algorithm

The Semi-Supervised Fuzzy Clustering with Multiple Kernels (SS-FCMK) minimizes the following objective function:

$$J = \sum_{i=1}^{C}\sum_{j=1}^{N}\sum_{k=1}^{M}\mu_{ij}^{2}\frac{w_{ik}^{2}}{\sigma_k}\left(1-\exp\left(-\frac{\|x_j-c_i\|^2}{2\sigma_k^2}\right)\right) + \gamma\left(\sum_{(x_j,x_h)\in ML}\sum_{i=1}^{C}\sum_{s=1,s\neq i}^{C}\mu_{ij}\mu_{sh} + \sum_{(x_j,x_h)\in CL}\sum_{i=1}^{C}\mu_{ij}\mu_{ih}\right), \tag{5.1}$$

subject to

$$\sum_{k=1}^{M} w_{ik} = 1,\ \forall i, \text{ and } w_{ik} \geq 0\ \forall i,k \tag{5.2}$$

and

$$\sum_{i=1}^{C}\mu_{ij} = 1,\ \forall j, \quad \mu_{ij} \geq 0\ \forall i,j \quad \text{and} \sum_{j=1}^{N}\mu_{ij} > 0,\ \forall i. \tag{5.3}$$

It is an extension of the FCMK algorithm that takes into account both the feature-based similarity between data points and the pairwise constraints. As in the FCMK objective function, the first term in (5.1) is the sum of squared distances to the prototypes weighted by constrained memberships. This term reinforces the compactness of the clusters. The second term in Equation (5.1) is composed of:

- the cost of violating the pairwise *must-link* constraints. The penalty corresponding to the presence of two such points in different clusters is weighted by the corresponding membership values;
- the cost of violating the pairwise *cannot-link* constraints. The penalty corresponding to the presence of two such points in a same cluster is weighted by their membership values.

In (5.1), the weight $\gamma \in (0,1)$ provides a way of specifying the relative importance of the *must-link* and *cannot-link* constraints compared to the sum of inter-cluster distances. In our approach, we fix it as the ratio of the number of constraints to the total number of points.

In order to optimize (5.1) with respect to w_{ik}, we assume that w_{ik}'s are independent from each other and reduce the optimization problem to C independent problems. That is, we convert the objective function in (5.1) to the following C simpler set of functions

$$J_i = \sum_{j=1}^{N}\sum_{k=1}^{M}\mu_{ij}^2\frac{w_{ik}^2}{\sigma_k}\left(1-\exp\left(-\frac{\|x_j-c_i\|^2}{2\sigma_k^2}\right)\right) + \gamma\left(\sum_{(x_j,x_h)\in ML}\sum_{s=1,s\neq i}^{C}\mu_{ij}\mu_{sh} + \sum_{(x_j,x_h)\in CL}\mu_{ij}\mu_{ih}\right), \tag{5.4}$$

for $i = 1,\ldots,C$. As the cost of violating the pairwise constraints does not depend on the resolution weights w_{ik} explicitly, setting the derivative of J_i with respect to w_{ik} gives the same update equation for w_{ik} as the FCMK algorithm, i.e., Equation (3.27).

Similarly, to optimize (5.4) with respect to cluster centers c_i, we take the derivative of J_i with respect to c_i and set it to zero. Since the partial supervision information does not depend on the cluster centers, we obtain the same update equation for c_i as the FCMK algorithm, i.e., Equation (3.18).

In order to optimize (5.1) with respect to the fuzzy memberships μ_{ij}, we apply Lagrange multipliers and obtain

$$J_\lambda = \sum_{i=1}^{C}\sum_{j=1}^{N}\mu_{ij}^2 D_{ij}^2 + \gamma\left(\sum_{(x_j,x_h)\in ML}\sum_{i=1}^{C}\sum_{s=1,s\neq i}^{C}\mu_{ij}\mu_{sh} + \sum_{(x_j,x_h)\in CL}\sum_{i=1}^{C}\mu_{ij}\mu_{ih}\right) - \sum_{j=1}^{N}\lambda_j\left(\sum_{i=1}^{C}\mu_{ij}-1\right), \tag{5.5}$$

where

$$D_{ij}^2 = \frac{w_{ik}^2}{\sigma_k}\left(1-\exp\left(-\frac{\|x_j-c_i\|^2}{2\sigma_k^2}\right)\right). \tag{5.6}$$

With fixed centers and resolution weights, the pair (λ_j, μ_{ij}) is an extremum of (5.5) only if $\partial J_\lambda / \partial \lambda_j = 0$ and $\partial J_\lambda / \partial \mu_{ij} = 0$. These conditions yield the following relationships:

$$\frac{\partial J_\lambda}{\partial \lambda_j} = \sum_{i=1}^{C} \mu_{ij} - 1 = 0 \; \forall j \tag{5.7}$$

and

$$\begin{aligned} \frac{\partial J_\lambda}{\partial \mu_{ij}} &= 2\mu_{ij} D_{ij}^2 - \lambda_j + \gamma \left(\sum_{(x_j, x_h) \in ML} \sum_{s=1, s \neq i}^{C} \mu_{sh} + \sum_{(x_j, x_h) \in CL} \mu_{ih} \right) \\ &= 0 \end{aligned} \tag{5.8}$$

Assuming that the membership values do not change significantly from an iteration to the next, solving (5.8) for μ_{ij} yields

$$\mu_{ij} = \frac{\lambda_j}{2D_{ij}^2} - \gamma \frac{\sum_{(x_j, x_h) \in ML} \sum_{s=1, s \neq i}^{C} \mu_{sh} + \sum_{(x_j, x_h) \in CL} \mu_{ih}}{2D_{ij}^2}. \tag{5.9}$$

Using the constraint that $\sum_{i=1}^{C} \mu_{ij} = 1$, for $j \in \{1, \ldots, N\}$, and solving (5.9) for λ_j, we obtain

$$\lambda_j = \frac{1}{\sum_{i=1}^{C} \frac{1}{2D_{ij}^2}} + \gamma \frac{\sum_{i=1}^{C} \frac{\sum_{(x_j, x_h) \in ML} \sum_{s=1, s \neq i}^{C} \mu_{sh} + \sum_{(x_j, x_h) \in CL} \mu_{ih}}{2D_{ij}^2}}{\sum_{i=1}^{C} \frac{1}{2D_{ij}^2}}. \tag{5.10}$$

Substituting Equation (5.10) in Equation (5.9), we obtain the following update equation for the membership of feature point x_j to cluster i:

$$\mu_{ij} = \mu_{ij}^{FCMK} + \mu_{ij}^{Constraints} \tag{5.11}$$

where μ_{ij}^{FCMK} is given by Equation (3.16), and

$$\mu_{ij}^{Constraints} = \frac{\gamma}{2D_{ij}^2} \left(\overline{Cv}_j - Cv_{ij} \right). \tag{5.12}$$

In Equation (5.12), Cv_{ij} and $\overline{Cv}_j$ are defined as

$$Cv_{ij} = \sum_{(x_j,x_h)\in ML} \sum_{s=1,s\neq i}^{C} \mu_{sh} + \sum_{(x_j,x_h)\in CL} \mu_{ih}, \tag{5.13}$$

and

$$\overline{Cv}_j = \frac{\sum_{i=1}^{C} \frac{\left(\sum_{(x_j,x_h)\in ML} \sum_{s=1,s\neq i}^{C} \mu_{sh} + \sum_{(x_j,x_h)\in CL} \mu_{ih}\right)}{D_{ij}^2}}{\sum_{i=1}^{C} \frac{1}{D_{ij}^2}}. \tag{5.14}$$

The first term in Equation (5.11) is the membership term in the FCMK algorithm and only focuses on distances between feature points and prototypes. The second term takes into account the available supervision: memberships are reinforced or reduced according to the pairwise constraints provided by the user. Cv_{ij} is the constraint violation cost associated with feature point x_j if it is assigned to cluster i, while $\overline{Cv}_j$ is the weighted average, over all the clusters, of the violation costs associated with feature point x_j. If the violated constraints do not involve feature point x_j, then $\mu_{ij}^{Constraints} = 0$.

If, by assigning point x_j to cluster i the violation cost is higher than the weighted average, it is intuitively better to reduce its membership in cluster i and this is exactly what the updating equation does ($\mu_{ij}^{Constraints}$ is a negative quantity in Equation (5.11)). Conversely, when the former cost is less than the weighted average, $\mu_{ij}^{Constraints}$ is a positive quantity and the membership value μ_{ij} will be increased.

The SS-FCMK algorithm is summarized in Algorithm 5.1.

Algorithm 5.1 The Semi-Supervised Fuzzy Clustering with Multiple Kernels (SS-FCMK)

1: Fix the number of clusters C, fuzzification parameter $m = 2$;
2: Initialize the fuzzy partition matrix **U**;
3: Initialize the cluster prototypes **C**;
4: Fix the number of kernels M and the scaling parameters $\sigma_1, \ldots, \sigma_M$;
5: Create $\boldsymbol{ML}$ and $\boldsymbol{CL}$ pairwise constraints;
6: **repeat**
7: Compute α_{ijk} coefficients using Equation (3.21);
8: Compute β_{ik} coefficients using Equation (3.24);
9: Update weights using Equation (3.27);
10: Compute the dissimilarity D_{ij} for all clusters using Equation (3.8);
11: Update fuzzy memberships using Equation (5.11);
12: Update cluster prototypes using Equation (3.18);
13: **until** (fuzzy memberships do not change or maximum number of iteration is reached).

Partial supervision information can also come from users feedback. In contrast to relevance feedback which asks users to label retrieved information, providing pairwise constraints does not necessarily require users to have prior knowledge or experience with the dataset. Typically, the system identifies the most ambiguous pairs of samples and presents them to the user. The user then provides the constraint information as a feedback. In order to maximize the utility of the limited supervised data available in a semi-supervised setting, pairwise constraints should be, if possible, actively selected as maximally informative ones rather than chosen at random. This would imply that fewer constraints will be required to significantly improve the clustering accuracy. To this end, pairwise constraints are selected among points lying at the clusters boundaries.

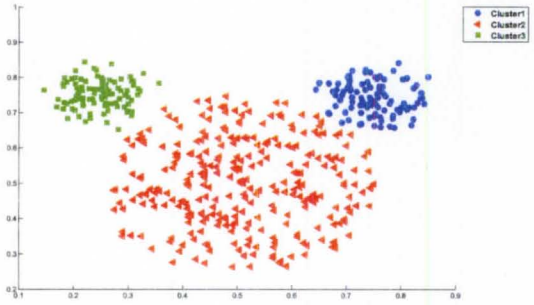
5.1.2 Performance illustration

To illustrate the ability of SS-FCMK to learn appropriate resolution weights and cluster the data simultaneously, we use it to categorize synthetic 2D datasets. We use the three datasets where the unsupervised FCMK did not perform well. These are dataset 2 (Figure (3.2(b))), dataset 3 (Figure (3.2(c))) and dataset 4 (Figure (3.2(d))). Dataset 1 is not considered because FCMK have partitioned it correctly. Thus, no further enhancement is possible. We use the same experiment setting as in Section 3.2.

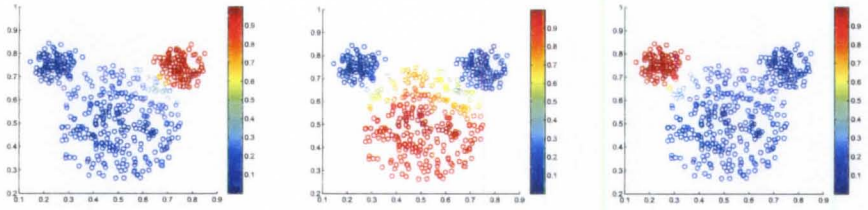
In order to construct the set of *must-link* and *cannot-link* constraints, we randomly select few points that are at the boundary of each cluster. For each dataset, we select 2% of the total number of points to construct the set of *must-link* and *cannot-link*. Pairs of selected points that belong to the same cluster (using the ground truth) constitute the *must-link* set, $\boldsymbol{ML}$. Similarly, pairs of points belonging to different clusters constitute the *cannot-link* set, $\boldsymbol{CL}$.

Table 5.1 reports the results of dataset 2, where three clusters are close to each others and have different densities and sizes. First, we notice that the learned resolution weights are close to the weights learned by FCMK (refer to Table 3.2). Second, the points at the boundaries of cluster 1 and cluster 2 are better categorized than for the FCMK approach reported in Table 3.2. As it can be seen in Figure (5.1), the learned fuzzy memberships of cluster 1 and cluster 2 are better characterized in comparison to the FCMK results (Figure (3.4)). Thus, the pairwise constraints are helpful in guiding the algorithm to learn better partition and more relevant resolution weights.

Table 5.1: Partition and resolution weights learned by SS-FCMK for dataset 2 displayed in Figure 3.2(b)



	$\sigma_k = 0.01$	$\sigma_k = 0.05$	$\sigma_k = 0.1$	$\sigma_k = 0.5$	True (σ_x, σ_y)
Cluster 1	0.316	0.328	0.198	0.158	$\sigma_{cl1} = (0.04; 0.04)$
Cluster 2	0.205	0.233	0.323	0.239	$\sigma_{cl2} = (0.12; 0.12)$
Cluster 3	0.317	0.327	0.198	0.158	$\sigma_{cl3} = (0.04; 0.04)$



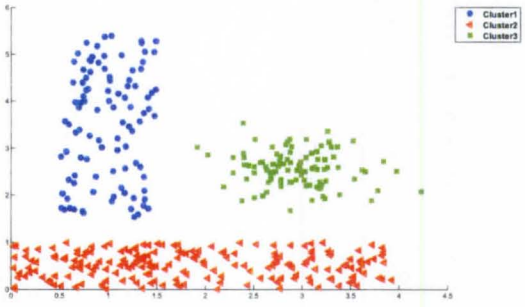
(a) (b) (c)

Figure 5.1: Fuzzy memberships learned by SS-FCMK on dataset 2 (Figure 3.2(b)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

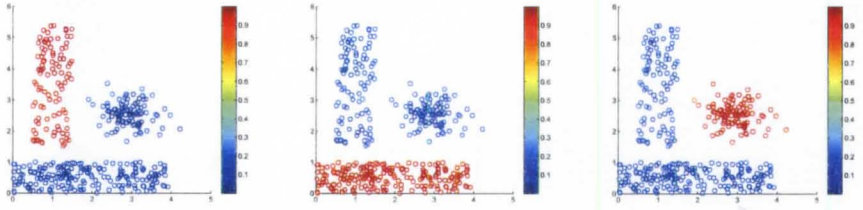
Table 5.2 displays the results of applying SS-FCMK on dataset 3. Compared to the FCMK results (refer to Table 3.3), this is a better partition. In fact, the points lying at the boundaries separating the different clusters are better categorized. This is also reflected in the learned fuzzy memberships as displayed in Figure (5.2). The improvement indicates that the supervision information is helpful in guiding the algorithm to learn better resolution weights.

As reported in Table 3.4, FCMK was not able to partition dataset 4 correctly. Using 2% of the data for partial supervision, the SS-FCMK succeeds to cluster this data as shown in Table 5.3. Moreover, the learned resolution weights reflect the multi-resolution of the data. This also reflected in the learned fuzzy memberships as displayed in Figure (5.3). The enhancement is due to the few constraints that guided the algorithm to learn better

Table 5.2: Partition and resolution weights learned by SS-FCMK for dataset 3 displayed in Figure 3.2(c)



	$\sigma_k = 0.16$	$\sigma_2 = 0.55$	$\sigma_3 = 0.85$	$\sigma_4 = 1.4$	True σ
Cluster 1	0.085	0.365	0.387	0.163	$\sigma_{cl1} = (0.29; 1.17)$
Cluster 2	0.084	0.365	0.386	0.165	$\sigma_{cl2} = (1.14; 0.28)$
Cluster 3	0.348	0.456	0.196	0	$\sigma_{cl3} = (0.41; 0.37)$



(a) (b) (c)

Figure 5.2: Fuzzy memberships learned by SS-FCMK on dataset 3 (Figure 3.2(c)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

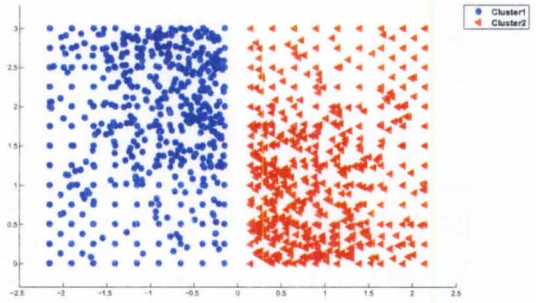
resolution weights.

5.2 Semi-supervised relational clustering with multiple kernels

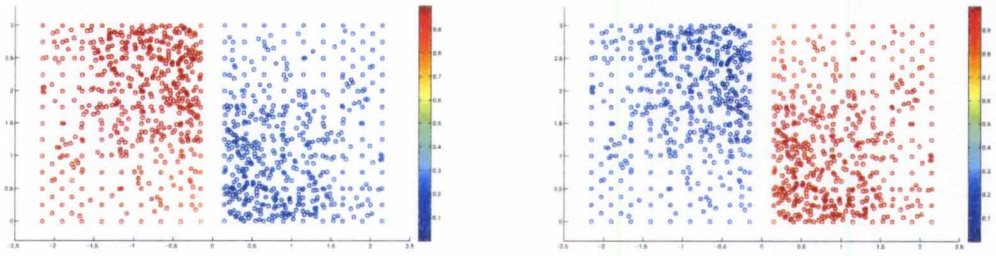
5.2.1 The semi-Supervised Relational Fuzzy C-Means with Multiple Kernels algorithm

The Semi-Supervised Relational Fuzzy C-means with Multiple Kernels (SS-RFCMK) is an extension of the RFCMK algorithm described in Chapter 4 that incorporates partial supervision information. It attempts to satisfy a set of *must-link* and *cannot-link* constraints

Table 5.3: Partition and resolution weights learned by SS-FCMK for dataset 4 displayed in Figure 3.2(d)



	$\sigma_k = 0.1$	$\sigma_k = 0.5$	$\sigma_k = 1$	$\sigma_k = 1.5$	True σ
Cluster 1	0.197	0.345	0.321	0.137	$\sigma_{cl1} = (0.56; 0.81)$
Cluster 2	0.193	0.341	0.335	0.131	$\sigma_{cl2} = (0.56; 0.81)$



(a) (b)

Figure 5.3: Fuzzy memberships learned by SS-FCMK on dataset 4 (Figure 3.2(d)) with respect to (a) cluster 1, and (b) cluster 2.

while minimizing the following objective function

$$J = \sum_{i=1}^{C} \frac{\sum_{j=1}^{N}\sum_{h=1}^{N} \mu_{ij}^2 \mu_{ih}^2 \widehat{r}_{jh}^{(i)}}{2\sum_{h=1}^{N} \mu_{ih}^2} + \gamma\left(\sum_{(x_j,x_h)\in ML} \sum_{i=1}^{C} \sum_{s=1,s\neq i}^{C} \mu_{ij}\mu_{sh} + \sum_{(x_j,x_h)\in CL} \sum_{i=1}^{C} \mu_{ij}\mu_{ih} \right), \tag{5.15}$$

subject to

$$\mu_{ij} \geq 0; \quad \sum_{i=1}^{C} \mu_{ij} = 1 \tag{5.16}$$

and

$$w_{ik} \in [0,1]\ \forall i,k; \quad \sum_{k=1}^{M} w_{ik} = 1\ \forall i. \tag{5.17}$$

The first term in (5.15) is the objective function of RFCMK. The second term consists of the cost of violating the pairwise *must-link* and *cannot-link* constraints. The penalty terms are weighted by the membership values of the points that violate the constraints. In other words, the penalty term is larger when the points are at the core of the cluster (high membership). When both terms are combined, SS-RFCMK will seek compact clusters and their resolution weights while minimizing the number of violated constraints. The value of γ in (5.15) controls the importance of the supervision compared to the sum of intra-cluster distances.

To optimize J with respect to the cluster membership $\mathbf{U}$, we use the Lagrange multiplier technique and obtain

$$\begin{aligned} J_\lambda = & \sum_{i=1}^{C} \frac{\sum_{j=1}^{N}\sum_{h=1}^{N} \mu_{ij}^2 \mu_{ih}^2 \left(\sum_{k=1}^{M} w_{ik}^2 \alpha_{jhk}\right)}{2\sum_{j=1}^{N} \mu_{ij}^2} \\ & + \gamma\left(\sum_{(x_j,x_h)\in ML}\sum_{i=1}^{C}\sum_{s=1,s\neq i}^{C} \mu_{ij}\mu_{sh} + \sum_{(x_j,x_h)\in CL}\sum_{i=1}^{C} \mu_{ij}\mu_{ih}\right) \\ & - \sum_{j=1}^{N} \lambda_j \left(\sum_{i=1}^{C} \mu_{ij} - 1\right), \end{aligned}$$

where α_{jhk} is as defined in (4.20).

By setting gradient of J to zero, we obtain,

$$\frac{\partial J_\lambda}{\partial \mu_{ij}} = \frac{2\mu_{ij}\sum_{h=1}^{N}\mu_{ih}^2\left(\sum_{k=1}^{M}w_{ik}^2\alpha_{jhk}\right)}{\sum_{j=1}^{N}\mu_{ij}^2} - \lambda_j - \frac{\mu_{ij}\sum_{j=1}^{N}\sum_{h=1}^{N}\mu_{ij}^2\mu_{ih}^2\left(\sum_{k=1}^{M}w_{ik}^2\alpha_{jhk}\right)}{\left(\sum_{j=1}^{N}\mu_{ij}^2\right)^2} + \gamma\left(\sum_{(x_j,x_h)\in ML}\sum_{s=1,s\neq i}^{C}\mu_{sh} + \sum_{(x_j,x_h)\in CL}\mu_{ih}\right) = 0 \tag{5.18}$$

Expanding (4.8), we obtain

$$D_{ij}^2 = \frac{\sum_{h=1}^{N}\mu_{ih}^2\left(\sum_{k=1}^{M}w_{ik}^2\alpha_{jhk}\right)}{\sum_{j=1}^{N}\mu_{ij}^2} - \frac{\sum_{j=1}^{N}\sum_{h=1}^{N}\mu_{ij}^2\mu_{ih}^2\left(\sum_{k=1}^{M}w_{ik}^2\alpha_{jhk}\right)}{2\left(\sum_{j=1}^{N}\mu_{ij}^2\right)^2}. \tag{5.19}$$

Using (5.19) and rearranging the terms in (5.18), we obtain

$$\mu_{ij} = \frac{\lambda_j}{2D_{ij}^2} - \gamma\frac{\sum_{(x_j,x_h)\in ML}\sum_{s=1,s\neq i}^{C}\mu_{sh} + \sum_{(x_j,x_h)\in CL}\mu_{ih}}{2D_{ij}^2}. \tag{5.20}$$

Using $\sum_{i=1}^{C}\mu_{ij} = 1$, and solving for λ_j, the update equation for the memberships reduces to

$$\mu_{ij} = \mu_{ij}^{RFCMK} + \mu_{ij}^{Constraints} \tag{5.21}$$

where

$$\mu_{ij}^{RFCMK} = \frac{1}{\sum_{t=1}^{C}\left(\frac{D_{ij}^2}{D_{tj}^2}\right)}, \tag{5.22}$$

$$\mu_{ij}^{Constraints} = \frac{\gamma}{2D_{ij}^2}\left(\overline{Cv}_j - Cv_{ij}\right). \tag{5.23}$$

In (5.23), Cv_{ij} and $\overline{Cv}_j$ are defined as

$$Cv_{ij} = \sum_{(x_j,x_h)\in ML} \sum_{s=1,s\neq i}^{C} \mu_{sh} + \sum_{(x_j,x_h)\in CL} \mu_{ih}, \tag{5.24}$$

and

$$\overline{Cv}_j = \frac{\sum_{i=1}^{C} \left(\frac{\sum_{(x_j,x_h)\in ML} \sum_{s=1,s\neq i}^{C} \mu_{sh} + \sum_{(x_j,x_h)\in CL} \mu_{ih}}{D_{ij}^2} \right)}{\sum_{i=1}^{C} \left(\frac{1}{D_{ij}^2} \right)}. \tag{5.25}$$

In other words, the membership of a point in a given cluster depends on its relative proximity to that cluster (μ^{RFCMK} term) and the cost of constraint violations incurred by that cluster assignment ($\mu^{Constraints}$ term).

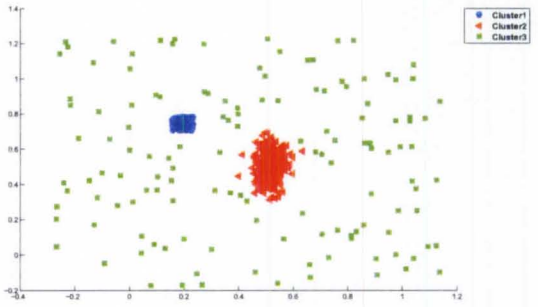
Since the pairwise constraints in (5.15) do not depend on the resolution weights w_{ik} explicitly, optimization of (5.15) with respect to w_{ik} yields the same update equation as in RFCMK, i.e., Equation (4.25).

The SS-RFCMK algorithm is summarized in Algorithm 5.2.

Algorithm 5.2 The Semi-Supervised Relational Fuzzy Clustering with Multiple Kernels (SS-RFCMK)

1: Fix the number of clusters C, the fuzzifier $m = 2$, the number of kernels M and kernel parameters $\sigma_1, \ldots, \sigma_M$;
2: Initialize the fuzzy partition matrix U;
3: Initialize $\beta = 0$, and $w_{ik} = 1/M$;
4: Create $\boldsymbol{ML}$ and $\boldsymbol{CL}$ pairwise constraints;
5: **repeat**
6: Compute total dissimilarities $\widehat{\mathbf{R}}^{(i)}$ using (4.5);
7: Compute $\widehat{\mathbf{R}}_{\beta}^{(i)}$ using (4.17);
8: Compute membership vectors v_i using (5.22);
9: Compute distances using (5.19);
10: **if** $D_{ij}^2 < 0$ for any i, j **then**
11: Compute $\Delta\beta$ using (2.19);
12: $D_{ij}^2 = D_{ij}^2 + (\Delta\beta/2) * \|v_i - e_j\|$;
13: $\beta = \beta + \Delta\beta$;
14: **end if**
15: Update fuzzy memberships using (5.21);
16: Update kernel combination weights using (4.25);
17: **until** (fuzzy memberships do not change).

Table 5.4: Partition and resolution weights learned by SS-RFCMK for dataset 6 displayed in Figure (4.1(b))



	$\sigma_k = 0.02$	$\sigma_k = 0.05$	$\sigma_k = 0.1$	$\sigma_k = 0.5$	True σ
Cluster 1	0.502	0.379	0.092	0.027	$\sigma_{cl1} = 0.02$
Cluster 2	0.094	0.319	0.502	0.085	$\sigma_{cl2} = 0.06$
Cluster 3	0.032	0.092	0.135	0.741	$\sigma_{cl3} = 0.35$

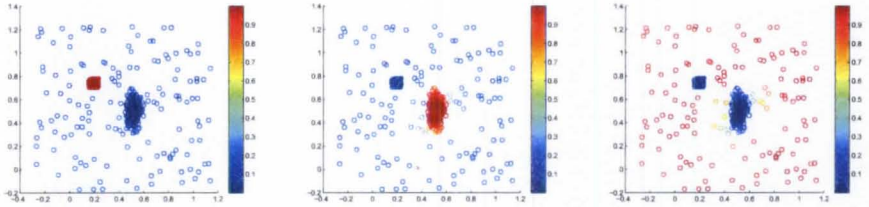
5.2.2 Performance illustration

To illustrate the ability of SS-RFCMK to use partial supervision information to improve the results, we use it to categorize dataset 6 displayed in Figure (4.1(b)), where the unsupervised RFCMK did not perform well. We use the same experimental setting as outlined in Section 4.2.

In order to construct the set of *must-link* and *cannot-link* constraints, we randomly select few points that are at the boundaries of each cluster. We select 2% of the total number of points to construct the set of *must-link* and *cannot-link*. Pairs of selected points that belong to the same cluster (using the ground truth) constitute *must-link* set, $\boldsymbol{ML}$. Similarly, pairs of points belonging to different clusters constitute *cannot-link* set, $\boldsymbol{CL}$.

Table 5.4 reports the results of dataset 6, where the three clusters are close to each others with unbalanced sizes and densities. We notice that the points at the boundaries of cluster 2 and cluster 3 are better categorized compared to the results of the RFCMK approach (refer to Table 4.2). Moreover, in terms of the fuzzy memberships, we can see from Figure (5.4) that the three clusters are better characterized compared to the RFCMK results (Figure (4.3)). Thus, the pairwise constraints help the clustering process to obtain a better partition.

In this chapter, we presented two semi-supervised algorithms that learn multiple kernels



(a) (b) (c)

Figure 5.4: Fuzzy memberships learned by SS-RFCMK on dataset 6 (Figure (4.1(b))) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

and the fuzzy partitioning of the data simultaneously, guided by a small amount of pairwise constraints. We have shown that the incorporation of these constraints have guided the clustering process to better learn the resolution weights and the fuzzy memberships that reflect the structure of the data. Consequently, a better partition of the data can be obtained. These pairwise constraints can be even more useful on real high dimensional datasets where the algorithm is more susceptible to local minima. Results on real and high-dimensional datasets will be reported in Chapter 7 and compared to other similar algorithms.

CHAPTER 6

RELATIONAL FUZZY CLUSTERING WITH MULTIPLE KERNELS FOR VERY LARGE DATA

The RFCMK algorithm (Chapter 4) has proven to be more effective than the FCMK approach (Chapter 3) to cluster complex data sets. However, RFCMK does not scale well to Very Large data (data that cannot be loaded into the computer's working memory). For instance, RFCMK has a memory requirement of $\mathcal{O}(N^2)$ to store the full NxN kernel matrix $K^{(i)}$, where N is the number of objects in the data set. Thus, even data sets that have nearly $1,000,000$ objects require terabytes of working memory which is impractical for most computers.

To overcome this limitation, in this thesis, we introduce two techniques aimed to extend our RFCMK algorithm presented in Chapter 4 to cluster very large data. The random sample and extend RFCMK (rseRFCMK) is based on sampling followed by non-iterative extension. The single pass RFCMK (spRFCMK) is an incremental technique that makes one sequential pass through subsets of the data. Both rseFCMK and spFCMK strive to find a good partitioning of the data into meaningful clusters and the optimal kernel-induced feature map. The main challenge arising from RFCMK for VL data is that both the optimal clustering and the kernel combinations need to be learned simultaneously. More importantly, the solutions to kernel clustering and multiple kernel combinations

are strongly correlated, making it a significantly more challenging problem than a typical incremental learning problem.

6.1 RFCMK algorithms for Very Large Data

6.1.1 Weighted RFCMK

First, we generalize the RFCMK algorithm by introducing a weight for each point x_j, denoted by γ_j. The resulting algorithm, called weighted RFCMK (wRFCMK), introduces weights that define the relative importance of each object in the clustering solution, similar to the wFCM [68].

The weighted RFCMK (wRFCMK) modifies the objective function of the RFCMK (see Equation (4.2)) to the following:

$$J = \sum_{i=1}^{C} \frac{\sum_{j=1}^{N}\sum_{h=1}^{N} \mu_{ij}^{m}\mu_{ih}^{m}\gamma_j\gamma_h\|\psi^{(i)}(x_j)-\psi^{(i)}(x_h)\|^2}{2\sum_{h=1}^{N}\gamma_h\mu_{ih}^{m}}, \tag{6.1}$$

where $\gamma \in \mathbb{R}^N$, $\gamma_j \geq 0\ \forall j$, is a set of weights for each feature vector.

The cluster centers $\psi^{(i)}(c_i)$ in feature space can be defined as linear combinations of the feature vectors, i.e.,

$$\psi^{(i)}(c_i) = \frac{\sum_{h=1}^{N}\mu_{ih}^{m}\psi^{(i)}(x_h)}{\sum_{h=1}^{N}\mu_{ih}^{m}}. \tag{6.2}$$

Since we assume that each object $\psi^{(i)}(x_j)$ has a different predetermined influence, given by a respective weight γ_j, the centers in (6.2) become

$$\psi^{(i)}(c_i) = \frac{\sum_{h=1}^{N}\gamma_h\mu_{ih}^{m}\psi^{(i)}(x_h)}{\sum_{h=1}^{N}\mu_{ih}^{m}}. \tag{6.3}$$

The Euclidean distance from feature point x_j to center c_i in the kernel space is given by

$$D_{ij}^2 = K^{(i)}(x_j,x_j) - 2\frac{\sum_{h=1}^{N}\mu_{ih}^{m}\gamma_h K^{(i)}(x_j,x_h)}{\sum_{h=1}^{N}\mu_{ih}^{m}\gamma_h} + \frac{\sum_{h=1}^{N}\sum_{h'=1}^{N}\mu_{ih}^{m}\mu_{ih'}^{m}\gamma_h\gamma_{h'}K^{(i)}(x_h,x_{h'})}{\left(\sum_{h=1}^{N}\mu_{ih}^{m}\gamma_h\right)^2}, \tag{6.4}$$

where the α_{ijk} coefficients are defined as

$$\alpha_{ijk} = K_k(x_j, x_j) - \frac{2\sum_{h=1}^{N} \mu_{ih}^m \gamma_h K_k(x_j, x_h)}{\sum_{h=1}^{N} \mu_{ih}^m \gamma_h} + \frac{\sum_{h=1}^{N}\sum_{h'=1}^{N} \mu_{ih}^m \mu_{ih'}^m \gamma_h \gamma_{h'} K_k(x_h, x_{h'})}{\left(\sum_{h=1}^{N} \mu_{ih}^m \gamma_h\right)^2}. \tag{6.5}$$

The wRFCMK algorithm is outlined in Algorithm 6.1.

Algorithm 6.1 The wFCMK algorithm

1: Fix the number of clusters C, $m > 1$, the number of kernels M, the scaling parameters $\sigma_1, \ldots, \sigma_M$ and the weights for each point γ_j;
2: Initialize the fuzzy partition matrix U;
3: **repeat**
4: Compute coefficients α_{ijk} using (6.5);
5: Compute kernel weights w_{ik} using (4.25);
6: Compute distances using (6.4);
7: Update fuzzy memberships using (5.22);
8: **until** (fuzzy memberships do not change);
9: Compute cluster prototypes $P_i = \arg\min_j(D_{ij}^2)$.

As it can be seen in Algorithm 6.1, wRFCMK outputs the index of the object closest to each cluster center, which is called the cluster prototype. The vector of indices P is important in the VL data schemes that are now proposed.

6.1.2 Random Sample and Extend RFCMK

Similar to geFFCMK [16], we propose the random sample and extend RFCMK (rseRFCMK) approach. A sample X_s of the data X is chosen, and this sample is clustered using wRFCMK. The cluster prototypes that are returned by wRFCMK are then used to extend the partition to the entire dataset using

$$D_{ij}^2 = K^{(i)}(x_j, x_j) + K^{(i)}(x_{P_i}, x_{P_i}) - 2K^{(i)}(x_j, x_{P_i}). \tag{6.6}$$

In contrast to geFFCM that uses progressive sampling to draw a sample that is representative of the full dataset, we believe that random sampling is sufficient for VL data and computationally less expensive [34]. In fact, for VL data, this representative sample may be large itself. Thus, we randomly draw without replacement a predetermined sample of the data.

Since RFCMK does not provide cluster centers, we cannot directly apply the sample and extend approach as in geFFCM. Therefore, we use the wRFCMK algorithm to cluster the first random sample and return a set of cluster prototypes P. The prototypes are the indices of the C objects that are the closest to the cluster centers in the kernel space.

The rseFCMK algorithm is summarized in Algorithm 6.2.

Algorithm 6.2 The rseFCMK algorithm

1: Fix the number of clusters C, $m > 1$, the number of kernels M and the scaling parameters $\sigma_1, \ldots, \sigma_M$;
2: Sample N_s vectors from X, denoted X_s;
3: Compute kernel weights and prototypes P of the chunk X_s using Algorithm 6.1 with $\gamma = \mathbf{1}_{N_s}$;
4: Compute distances using (6.6)
5: Compute fuzzy membership using (5.22);

The rseRFCMK algorithm is scalable. In fact, we can choose a sample size N_s to suit the computational resources available. However, the clustering iterations are not performed on the entire dataset. Hence, if the sample X_s is not representative of the full dataset, then rseRFCMK cannot accurately approximate RFCMK solution. Therefore, we need an approach that operates on the full dataset by separating it into multiple chunks.

6.1.3 Single Pass RFCMK

Based on the SPFCM algorithm [68], we propose the single pass RFCMK (spRFCMK) algorithm. First, the data is divided into chunks. Then, the first chunk is loaded into memory and clustered into C partitions. The data in memory is condensed into C weighted points before purging the memory. In each partial data access, a new chunk is loaded into memory and clustered along with the past C weighted points obtained from the previous clustering. This continues until all the data has been scanned once.

The single pass RFCMK (spRFCMK) which is outlined in Algorithm 6.3 follows the same idea as SPFCM. At Line 2, we randomly draw without replacement $s(N/s)$-sized samples X_l of X. The sample X_1 is the set of objects to be clustered in the first step of the algorithm. At Line 3, these objects are clustered with unity-valued weights. At Line 4, the weights for the C cluster prototypes are computed. Lines 5–10 comprise the main loop of spRFCMK. At Line 6, a weight vector of size $(N/s+C)$ is created, which includes the C weights of the cluster prototypes returned by the previous iteration and N/s 1s. At Line 8 the objects are

clustered using wRFCMK where the size of the kernel matrix $K^{(i)}$ is $(N/s+C)$x$(N/s+C)$. Finally, at Line 9 the weights of the C new prototypes are computed.

Algorithm 6.3 The spFCMK algorithm

1: Fix the number of clusters C, $m > 1$, the number of kernels M, the scaling parameters $\sigma_1, \ldots, \sigma_M$ and the number of chunks s;
2: Randomly draw s (approximately) equal sized subsets from X, denoted $\{X_1, \ldots, X_s\}$ with N_l the size of X_l;
3: Compute fuzzy membership μ_{ij}, kernel weights and prototypes P for X_1 using Algorithm 6.1 with $\gamma = \mathbf{1}_{N_1}$;
4: Compute the weights of the C cluster prototypes using $\gamma_i' = \sum_{j=1}^{N_1} \mu_{ij} \ \forall i$;
5: **for** $l = 2$ to s **do**
6: Create a vector of weights $\gamma = \{\gamma', \mathbf{1}_{N_l}\}$;
7: Combine the C cluster prototypes and the l^{th} chunk, denoted $E = \{P, X_l\}$;
8: $U, P, W = wRFCMK(C, m, [\sigma_1, \ldots, \sigma_M], \gamma)$;
9: Update the weights of the cluster prototypes, $\gamma_i' = \sum_{j=1}^{N_l+C} \mu_{ij} \ \forall i$;
10: **end for**

We should note that the cluster prototypes are weighted in wRFCMK by the sum of the respective memberships, i.e., $\sum_{j=1}^{N_l+C} \mu_{ij}$. Essentially, the weight causes the cluster prototypes to have more influence on the clustering solution than the data in the chunk newly loaded. For instance, each prototype represents the multiple data points in each cluster.

6.2 Complexity

We estimate the time and space complexity of each of the proposed VL variants of RFCMK. All operations and storage space are counted as unit costs. We do not assume economies that might be realized by special programming tricks or properties of the equations involved. Importantly, however, the asymptotic estimates that are shown in Table 6.1 for the growth in time and space with N, which is the number of objects in data X, are unaffected by changes in counting procedures.

Table 6.1 shows the complexities of the VL RFCMK algorithms in terms of problem variables: N is the number of objects in the p-dimensional data, $X \in \mathbb{R}^p$, C is the number of clusters, t is the number of iterations required for termination, M is the number of kernels, and s is the number of subsets that X is divided into by random sampling without replacement.

As it can be seen it Table 6.1, the main drawback of RFCMK is the $\mathcal{O}(N^2)$ memory requirement for the storage of the kernel matrix. The rseRFCMK algorithm combats this

Table 6.1: Time and space complexity of RFCMK for VL algorithms

Algorithm	Time	Space
wRFCMK, RFCMK	$\mathcal{O}(tCMN^2)$	$\mathcal{O}(N^2)$
rseRFCMK	$\mathcal{O}(tCMN^2/s^2)$	$\mathcal{O}(N^2/s^2)$
spRFCMK	$\mathcal{O}(tCMN^2/s)$	$\mathcal{O}(N^2/s^2)$

by only computing the $(N/s) \times (N/s)$ kernel matrix for the sampled data, resulting in an $\mathcal{O}(N^2/s^2)$ space complexity. On the other hand, The spRFCMK algorithm operates on $\mathcal{O}(N^2/s^2)$-sized kernel matrices.

The time complexity of the RFCMK is dominated by the the computation of (4.20). This calculation requires $\mathcal{O}(N^2)$ operations per cluster, resulting in a total time complexity of $\mathcal{O}(tCMN^2)$ for RFCMK and wRFCMK. The rseRFCMK algorithm is equivalent to RFCMK or wRFCMK for an (N/s)-sized dataset, resulting in a time complexity of $\mathcal{O}(tCMN^2/s^2)$. The proposed spRFCMK algorithm runs wRFCMK on s chunks of approximately (N/s)-sized data, which results in $\mathcal{O}(tCMN^2/s^2)$ time complexity.

In this chapter, we extended the RFCMK clustering to Very Large data. Specially, we implemented methods that are based on sampling followed by non-iterative extension, and incremental technique that makes one sequential pass through subsets of the data. Results on real and VL data sets will be reported in the next chapter and compared to other similar algorithms.

CHAPTER 7

EXPERIMENTAL RESULTS

The objective of the proposed algorithms (FCMK, RFCMK, SS-FCMK, and SS-RFCMK) is to partition the data and learn appropriate local density fitting functions. In this chapter, we assess the performance of the proposed approaches and compare their results to different clustering algorithms. First, we compare these algorithms using the same synthetic datasets used to illustrate FCMK and RFCMK (refer to Figures (3.2) and (4.1)). Then, in order to illustrate the ability of the proposed algorithms to learn cluster-dependent multiple kernels and to cluster real and high dimensional data, we use it to categorize a subset of the COREL image database and the handwritten digits database. A description of these datasets and the performance measures used to compare the different algorithms is provided in the following sections.

7.1 Data sets and performance measures

7.1.1 Image database

We use a subset of 500 color images from the COREL image collection. This subset includes five categories. Table (7.1) displays a sample image from each category along with the number of images for each category. The categories and the images within each one are selected to have different sizes and variances. We clustered this data set into five clusters. To compare the content of the images, we use two standard MPEG-7 descriptors [20].

Table 7.1: Sample image from each category of the COREL image database and the number of images for each category

Category	Antelopes	Beaches	Butterflies	Buses	Flowers
Sample image					
No. of images	100	100	100	100	100

- The Color Structure Descriptor (CSD) expresses local color structure in an image using an 8x8-structuring element. It counts the number of times a particular color is contained within the structuring element as the structuring element scans the image. The HMMD color space is used in this descriptor. The CSD consists of 32 features.

- The Edge Histogram Descriptor (EHD) represents the frequency and directionality of the edges within the image. First, simple edge detector operators are used to identify edges and group them into five categories: vertical, horizontal, 45-degree diagonal, 135-degree diagonal, and isotropic (non-edge). Then, local, global, and semi-local edge histograms are generated. The EHD feature is represented by a 150-dimensional vector.

7.1.2 Handwritten digits

The Pen-Based Recognition of Handwritten Digits dataset (Pen Digits) consists of 3498 examples [114] available at the UCI Machine Learning repository. The handwritten samples were collected using a pressure sensitive tablet that sends x and y coordinates of the pen at fixed time intervals. The raw data is processed so that each digit is represented by 8(x,y) coordinates resulting in a 16 dimensional feature vector. The classes correspond to the ten digits 0 to 9.

We randomly chose a subset of 1166 instances from the Pen Digits dataset. The categories within this data have different sizes and densities. Table (7.2) displays the number of samples from each category. We clustered this data set into 10 clusters.

Table 7.2: Number of samples from each category of the handwritten digits

Category	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
No. of samples	128	131	107	122	108	119	114	117	109	111

7.1.3 Performance measures

We assume that the ground truth is known. One way to assess the performance of the proposed approaches is to compute the classification rate. First, each cluster is assigned a label based on the majority of the true labels of its elements. Then, the correct classification rate of each cluster is computed. The overall accuracy (Q_{RR}) of the partition is computed as the average of the individual cluster rates weighted by cluster cardinality. Another way to evaluate the performance of a clustering algorithm is by comparing its partition matrix U to the ground truth partition matrix $U^{(T)} = [\mu_{ij}^{(T)}]$, where $\mu_{ij}^{(T)} \in \{0, 1\}$. The solution to the above problem is not trivial since the best one-to-one mapping of the clusters needs to be identified by finding the best permutation of the rows of the U matrices. In the following, we outline an efficient way for comparing the partition matrices indirectly based on the coincidence matrix, also called cluster connectivity matrix [60]. The coincidence matrices $\Psi^{(1)}$ and $\Psi^{(2)}$ of two partition matrices $U^{(1)}$ and $U^{(2)}$ are defined as

$$\Psi = [\psi_{jk}],\ 1 \leq j, k \leq N, \tag{7.1}$$

where

$$\psi_{jk} = \sum_{i=1}^{N} \mu_{ij}\mu_{ik}. \tag{7.2}$$

Using the two coincidence matrices, we compute a 2x2 contingency table as shown in Table (7.3) where

Table 7.3: Contingency table

	$\psi_{jk}^{(2)}=1$	$\psi_{jk}^{(2)}=0$	$\sum$
$\psi_{jk}^{(1)}=1$	N_{SS}	N_{SD}	$N_{S.}$
$\psi_{jk}^{(1)}=0$	N_{DS}	N_{DD}	$N_{D.}$
$\sum$	$N_{.S}$	$N_{.D}$	$N_{..}$

$$N_{SS}(\Psi^{(1)},\Psi^{(2)})=\sum_{j=2}^{N}\sum_{k=1}^{j-1}\psi_{jk}^{(1)}\psi_{jk}^{(2)}, \tag{7.3}$$

$$N_{SD}(\Psi^{(1)},\Psi^{(2)})=\sum_{j=2}^{N}\sum_{k=1}^{j-1}\psi_{jk}^{(1)}(1-\psi_{jk}^{(2)}), \tag{7.4}$$

$$N_{DS}(\Psi^{(1)},\Psi^{(2)})=\sum_{j=2}^{N}\sum_{k=1}^{j-1}(1-\psi_{jk}^{(1)})\psi_{jk}^{(2)}, \tag{7.5}$$

$$N_{DD}(\Psi^{(1)},\Psi^{(2)})=\sum_{j=2}^{N}\sum_{k=1}^{j-1}(1-\psi_{jk}^{(1)})(1-\psi_{jk}^{(2)}). \tag{7.6}$$

In the above, the indices S and D stand for Same cluster and Different clusters respectively.

Using the contingency Table (7.3), many measures can be computed to compare two partitions [20]. In our analysis, we use the *Rand statistics* (Q_{RS}), *Jaccard coefficient* (Q_{JC}), *Folkes-Mallows index* (Q_{FMI}), and the *Hubert index* (Q_{HI}) to compare each generated partition $U^{(cl)}$ to the ground truth partition $U^{(T)}$. These indices are defined as:

$$Q_{RS}(\Psi^{(cl)},\Psi^{(T)})=\frac{N_{SS}+N_{DD}}{N_{..}}, \tag{7.7}$$

$$Q_{JC}(\Psi^{(cl)},\Psi^{(T)})=\frac{N_{SS}}{N_{SS}+N_{SD}+N_{DS}}, \tag{7.8}$$

$$Q_{FMI}(\Psi^{(cl)},\Psi^{(T)})=\frac{N_{SS}}{\sqrt{(N_{SS}+N_{SD})(N_{SS}+N_{DS})}}, \tag{7.9}$$

$$Q_{HI}(\Psi^{(cl)},\Psi^{(T)})=\frac{N_{..}N_{SS}-N_{S.}N_{.S}}{\sqrt{N_{S.}N_{.S}N_{D.}N_{.D}}}. \tag{7.10}$$

All of the above measures provide larger values when the two partitions are more similar.

7.2 Evaluation of the unsupervised clustering algorithms

In this section, we assess the performance of the proposed unsupervised clustering algorithms, FCMK and RFCMK, and compare them to the following approaches:

- FCM [15] using the Euclidean distance to compute the distance between feature point x_j and center c_i;
- DBSCAN [70] with the neighborhood parameter tuned from 1 to 20 by an increment of 1;
- kNERF [88] outlined in Subsection 2.3.4. We tune the scaling parameter between 0.01 and 100 with a step of 0.1;
- Spectral clustering [118] with the neighborhood parameter tuned from 1 to 20 by an increment of 1;
- MKFC [57] using the same scaling parameters as FCMK and RFCMK.

To compare the performance of the different clustering algorithms, we assume the ground truth is known and use the cluster evaluation measures described in Subsection (7.1.3).

7.2.1 Synthetic datasets

In Section 3.2 and Section 4.2, we have illustrated the ability of FCMK and RFCMK to perform density fitting and partition synthetic 2-D datasets displayed in Figure (3.2) and Figure (4.1). We also provided an interpretation of the learned resolution weights. In order to better assess the efficacy of FCMK and RFCMK on these datasets, we compare them to the clustering approaches mentioned above. For all algorithms, we set the number of clusters C to the true one (see Figure (3.2) and Figure (4.1)), the fuzzifier m to 1.7 and the maximum number of iterations to 100.

Figure (7.1) displays the clustering results of the different algorithms on dataset 1. As it can be seen, The FCM (Figure 7.1(a)), DBSCAN (Figure 7.1(b)), kNERF (Figure 7.1(c)) and MKFC (Figure 7.1(e)) were not able to categorize this dataset correctly. Although, kNERF is based on a non linear mapping of the input data, it was not able to categorize this data correctly. This is due mainly to the fact that one global scaling parameter is not sufficient when the input data includes clusters with different local statistics. MKFC,

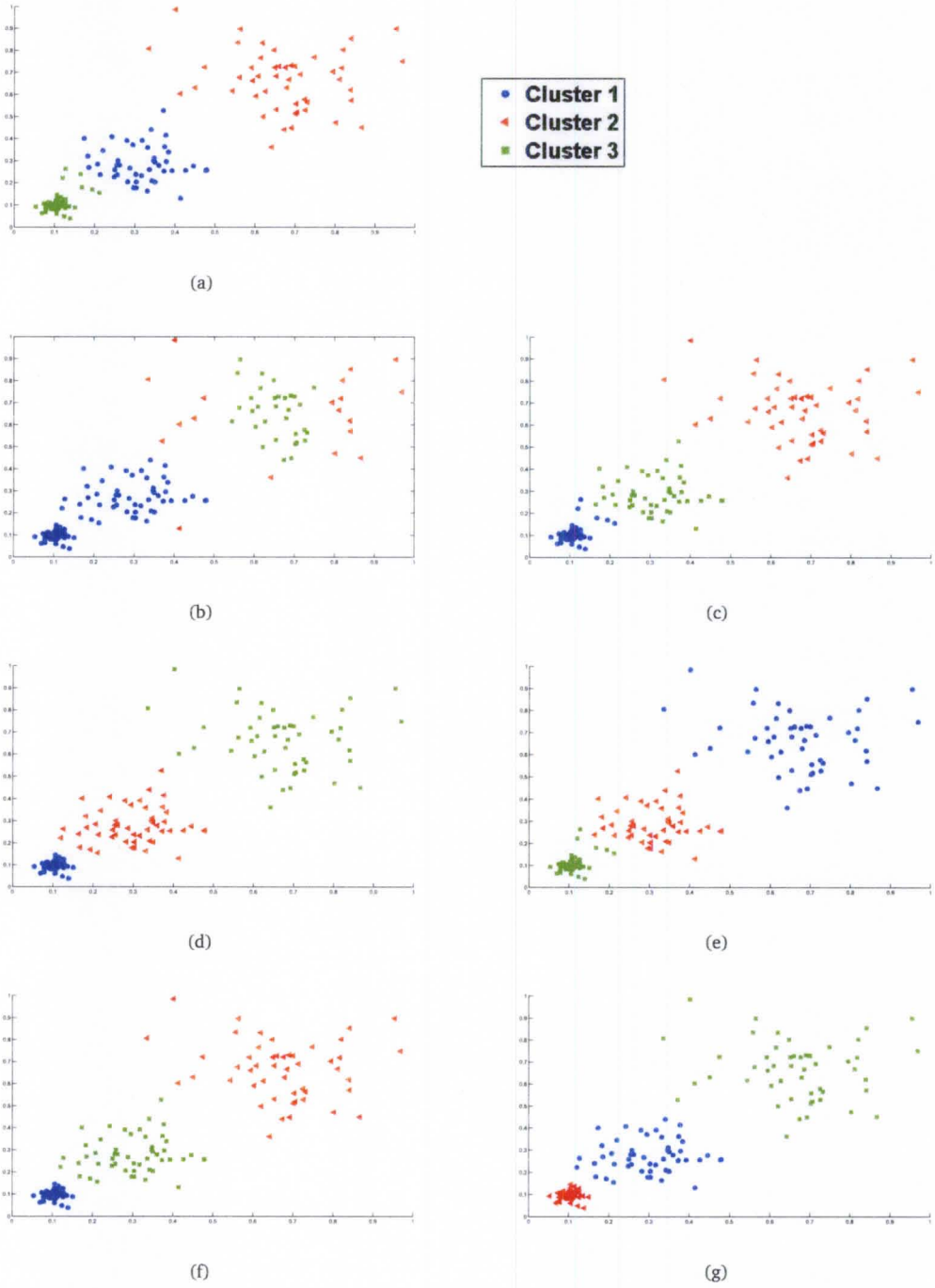


Figure 7.1: Results of clustering dataset 1 using (a) FCM, (b) DBSCAN, (c) kNERF, (d) Spectral, (e) MKFC, (f) FCMK, and (g) RFCMK

using a set of global kernel weights for all clusters, cannot categorize this data. This is because the one set of global kernel weights for all clusters cannot take into account the variations of the different clusters. Spectral was able to categorize dataset 1 correctly after tuning the neighboring parameters. On the other hand, FCMK and RFCMK were able to categorize this dataset without any parameter tuning.

Figure (7.2) displays the clustering results on dataset 2. As it ca be seen, neither FCM, DBSCAN, kNERF, Spectral or MKFC were able to categorize this data correctly. FCMK provides meaningful partition of this dataset. It has categorized few points at the boundaries incorrectly. On the other hand, RFCMK learned local density fitting and was able to partition this data correctly. This example illustrates the inability of existing algorithm to partition data that includes clusters close to each others with different sizes and densities.

Figure (7.3) displays the clustering results on dataset 3. The FCM algorithm cannot categorize this data as it can be seen in Figure 7.3(a). This is due to the fact that FCM is designed to seek compact spherical clusters. Similarly, the kNERF algorithm, using a single bandwidth, was not able to categorize this data correctly. We can notice that regardless of the variance of the distribution of the different clusters in the original feature space, kNERF approach maps all the points using the same global kernel-induced distance measure. This is not appropriate for this data as it can be seen in Figure 7.3(c). MKFC, using a set of global kernel weights for all clusters, cannot categorize this data as reported in Figure 7.3(e). DBSCAN and Spectral were able to partition this data correctly after tuning the neighboring parameters. FCMK provides meaningful partition of this dataset. It has categorized few points at the boundaries incorrectly. On the other hand, RFCMK was able to categorize the data without any tuning.

Figure (7.4) displays the clustering results on dataset 4. The FCM algorithm seeks compact spherical clusters ignoring potential multi-resolution within the same cluster. Thus, FCM cannot categorize properly this data set as it can be seen in Figure 7.4(a). The kNERF algorithm, using one uniform scaling parameter for all the data, was not able to categorize this data correctly. Although MKFC uses multiple kernels, the weight assigned to each kernel is spatially constant all over the implicit Hilbert space. Such treatment ignores the possible disparity of data structure in different spatial areas as seen in Figure 7.4(e). The FCMK algorithm was not able to categorize this data correctly. This is due

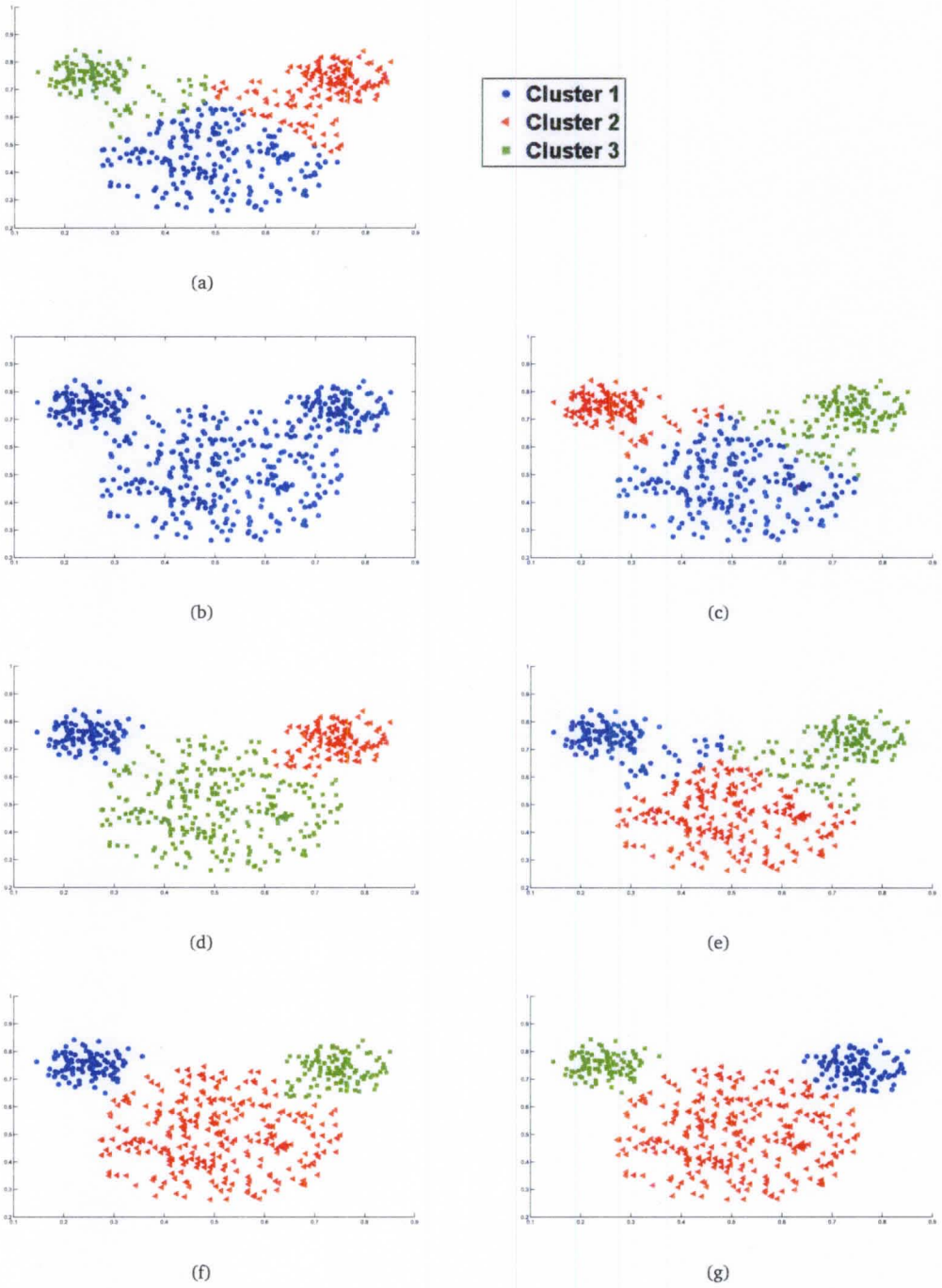


Figure 7.2: Results of clustering dataset 2 using (a) FCM, (b) DBSCAN, (c) kNERF, (d) Spectral, (e) MKFC, (f) FCMK, and (g) RFCMK

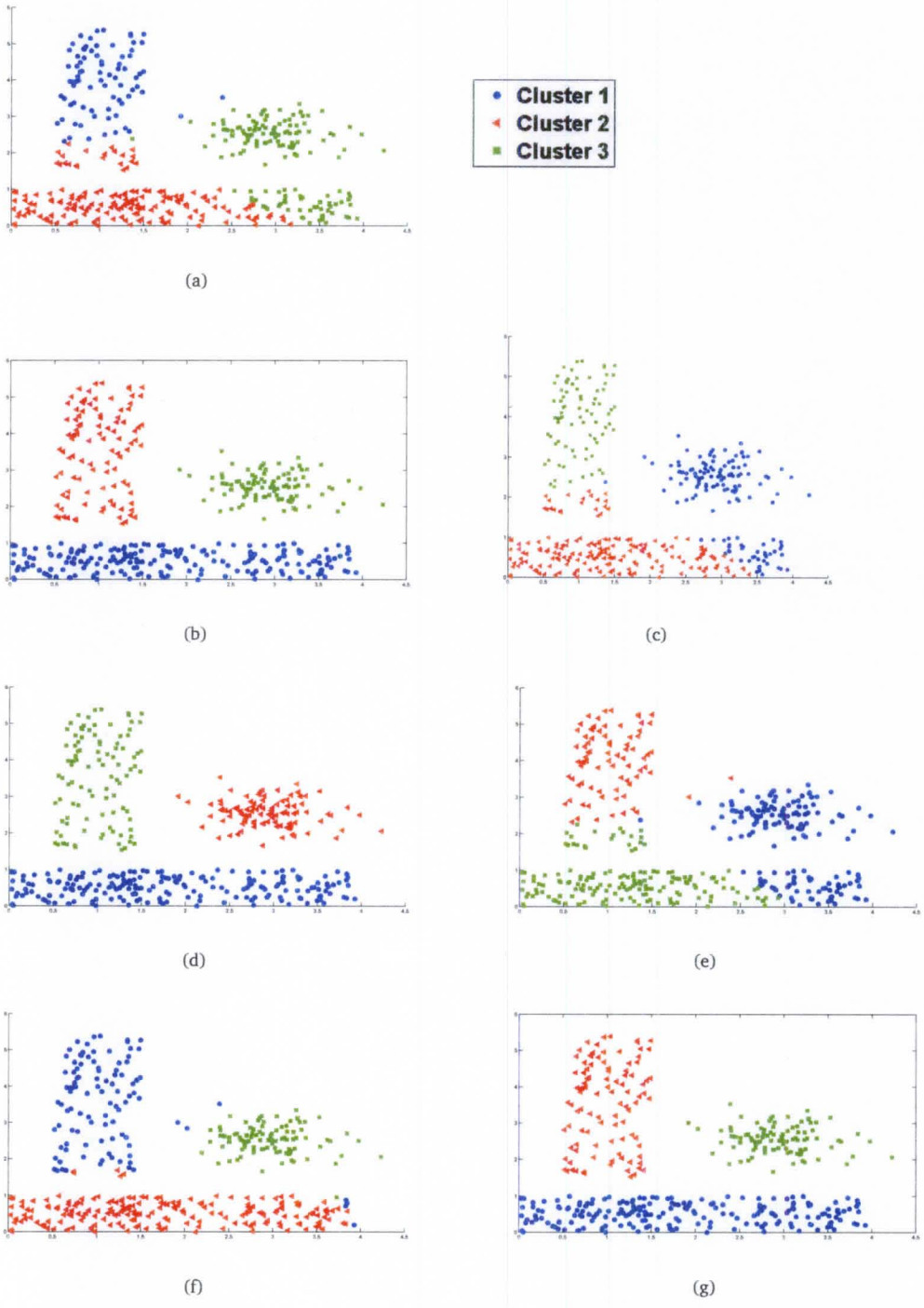


Figure 7.3: Results of clustering dataset 3 using (a) FCM, (b) DBSCAN, (c) kNERF, (d) Spectral, (e) MKFC, (f) FCMK, and (g) RFCMK

Figure 7.4: Results of clustering dataset 4 using (a) FCM, (b) DBSCAN, (c) kNERF, (d) Spectral, (e) MKFC, (f) FCMK, and (g) RFCMK

to the fact that FCMK estimates the density by counting the number of points within a specified radius. However, the dataset in Figure 3.2(d) contains clusters with large intra-cluster and small inter-cluster distances. DBSCAN and Spectral were able to partition this data correctly after tuning the neighboring parameters. On the other hand, RFCMK was able to categorize the data without any tuning. It also was able to distinguish the different intrinsic sub-structures of clusters thanks to the learned resolution weights.

Figure (7.5) displays the clustering results on dataset 5. The FCM algorithm seeks spherical clusters and thus, cannot categorize this data. Similarly, FCMK cannot categorize this data as it can be seen in Figure 7.5(f). This is because FCMK is a prototype-based method unable to partition data where groups of similar objects cannot be represented efficiently by a single prototype. On the other hand, DBSCAN, kNERF and Spectral provide correct partition of this dataset. The spectral algorithm was able to do this by tuning the neighborhood within a user specified range to find local scaling parameters. On the other hand, RFCMK without tuning any parameters learn the local resolution weights and performs clustering simultaneously.

Figure (7.6) displays the clustering results on dataset 6. The dataset includes two dense clusters with different sizes and a third large cluster that is very sparse. Only spectral and RFCMK were able to categorize this data correctly. This is due to the density fitting with respect to each cluster generated by RFCMK.

Figure (7.7) displays the clustering results of dataset 7. This dataset is constituted of two concentric ovals. It does not correspond to the classical way of perceiving intra-cluster distance and density. As a result, only spectral and RFCMK were able to categorize this data correctly. However, RFCMK was able to partition the data without fixing any parameter. The learned resolution weights provide a solution for parameter selection.

7.2.2 Application to image database categorization

To illustrate the ability of FCMK and RFCMK to learn appropriate local density fitting functions and cluster the data simultaneously, we use it to categorize a subset of the COREL image database described in Subsection 7.1.1. The feature vectors described in Subsection 7.1.1 are concatenated to construct a 182 dimensional feature vector. For all algorithms, we fix the number of clusters to five (since we have five categories) and the

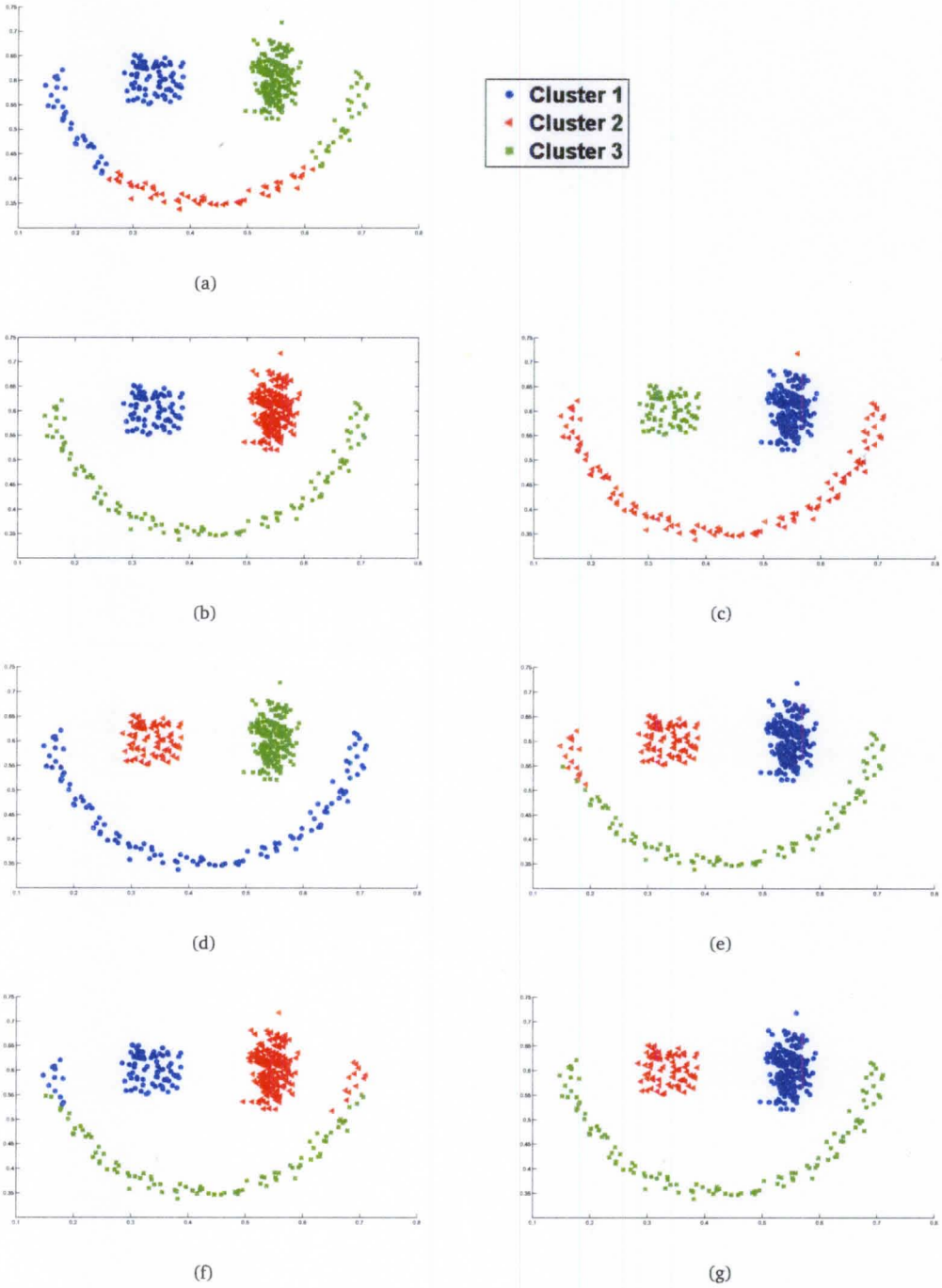


Figure 7.5: Results of clustering dataset 5 using (a) FCM, (b) DBSCAN, (c) kNERF, (d) Spectral, (e) MKFC, (f) FCMK, and (g) RFCMK

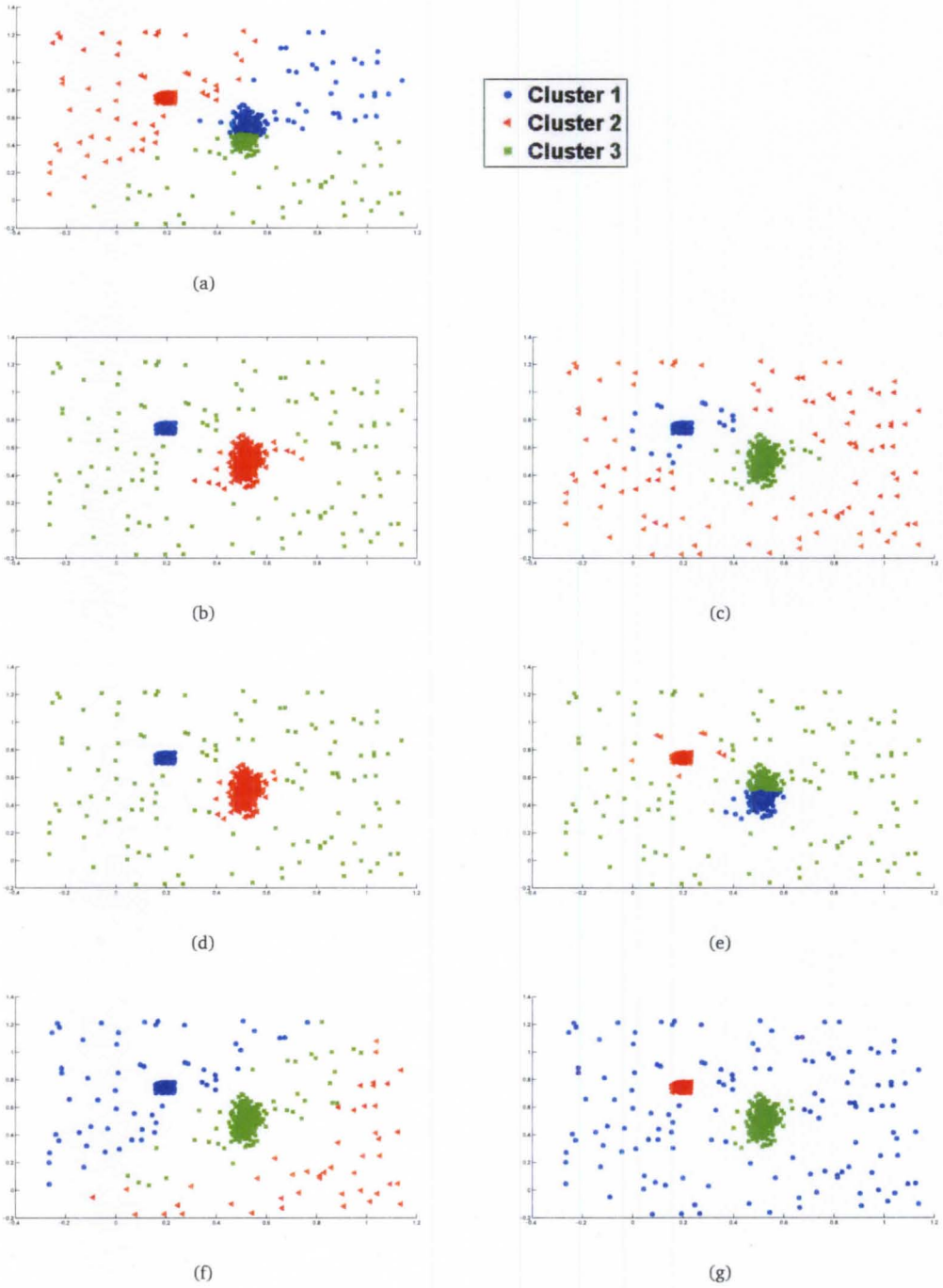


Figure 7.6: Results of clustering dataset 6 using (a) FCM, (b) DBSCAN, (c) kNERF, (d) Spectral, (e) MKFC, (f) FCMK, and (g) RFCMK

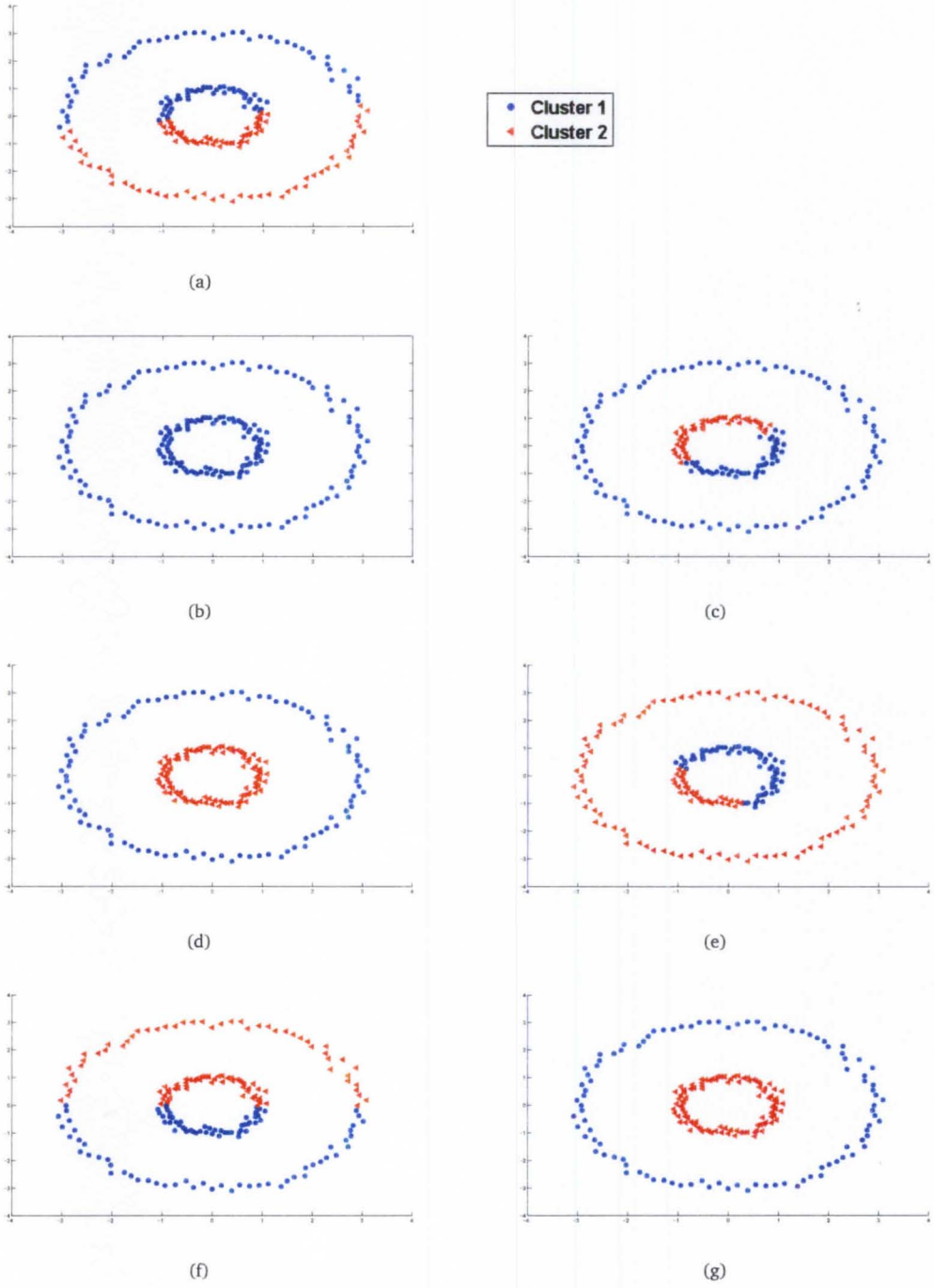


Figure 7.7: Results of clustering dataset 7 using (a) FCM, (b) DBSCAN, (c) kNERF, (d) Spectral, (e) MKFC, (f) FCMK, and (g) RFCMK

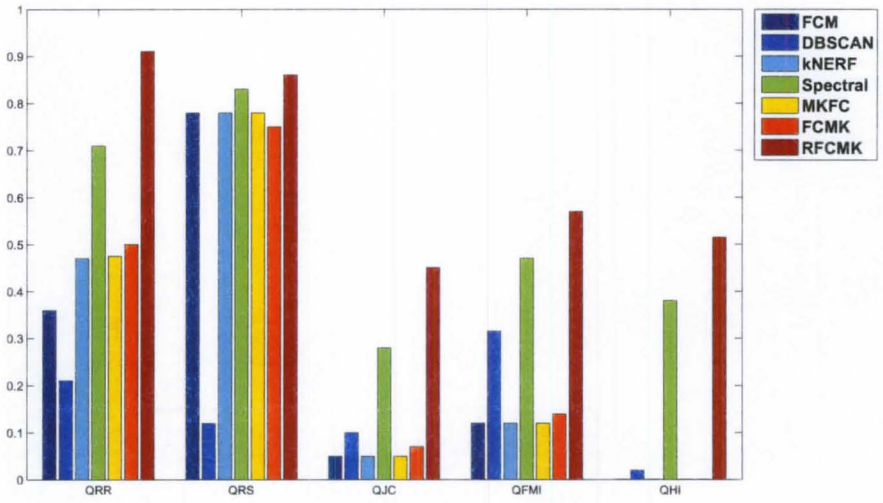


Figure 7.8: Performance measures obtained on categorizing COREL dataset using FCM, DBSCAN, kNERF, Spectral, MKFC, FCMK, and RFCMK clustering approaches.

fuzzifier m to 1.7. For DBSCAN, kNERF and spectral, we use the same strategy to tune their parameters as described in Section 7.2. Since we should not use the ground truth to select the optimal parameters for each algorithm, we select the parameter that gives the minimum intra-cluster dissimilarity.

To quantify the performance of the different clustering algorithms and compare them, we assume that the ground truth is known. We evaluate the performance of the algorithms using the Accuracy (Q_{RR}), Rand statistics (Q_{RS}), Jaccard coefficient (Q_{JC}), Folkes-Mallows index (Q_{FMI}), and the Hubert index (Q_{HI}) as described in Subsection 7.1.3.

Figure (7.8) compares the performance measures of the seven considered algorithms. As it can be seen, object-based algorithms (FCM and FCMK) cannot partition this data correctly. In fact, one prototype is not sufficient for sparse high dimensional spaces. Moreover, kNERF, spectral and DBSCAN were not able to categorize this image database as good as RFCMK. In fact, these algorithms are sensitive to the choice of their respective parameters. However, for high dimensional data it is not trivial to visualize the clustering results in order to choose the parameter that gives the best partition. Moreover, since we should not use the ground truth to compute the performance measures for the purpose of parameter selection, we selected the parameter that gives the minimum intra-cluster dissimilarity. Therefore, the choice of the parameters may not be optimal in this case. Although MKFC uses multiple kernels, it did not perform as well as RFCMK. This is

because it assigns constant weight to each kernel ignoring the possible disparity of data structure in different spatial areas. Figure (7.8) shows that RFCMK outperforms all of the other methods with respect to all five measures without tuning any parameter.

Since the spectral algorithm has the closest performance to RFCMK, we will use the partitions of these two algorithms to compare and analyze the results in more details. Tables 7.4 and 7.5 summarize the partitions obtained by the spectral and RFCMK algorithms. For each algorithm, we display one sample image from each cluster, the number of images assigned to the cluster, and the most representative images (i.e. images closest to the cluster center for spectral algorithm and images with high memberships for RFCMK). As it can be seen in Table 7.4, the spectral algorithm has combined several images from different categories into the same cluster. For instance, cluster 3 contains images from "Antelope" and "Butterfly" categories, and cluster 4 contains images from "Beach", "Butterfly" and "Antelope" categories. On the other hand, RFCMK was able to partition most of the data correctly because it adapted a kernel-induced similarity to each cluster.

The learned resolution weights are reported in Table 7.6 for each cluster. As it can be seen, cluster 2 ("Bus") has the smallest pairwise standard deviation of 0.078. Consequently, RFCMK assigns higher weight to $\sigma_k = 0.1$ for cluster 2. On the other hand, RFCMK assigns higher weight to $\sigma_k = 0.15$ for cluster 5 ("Flower") which is the closest to its pairwise standard deviation. Cluster 1 ("Butterfly"), cluster 3 ("Antelope") and cluster 4 ("Beach") have relatively close pairwise standard deviation. RFCMK was able to identify cluster 3 ("Antelope") by learning higher weights for $\sigma_k = 0.2$ and $\sigma_k = 0.5$. A higher weight for $\sigma_k = 0.5$ prevents images that are spatially close but, not within the dense region, from being assigned to this cluster.

7.2.3 Application to categorization of handwritten digits

In this experiment, we compare the performance of the considered clustering algorithms to categorize the handwritten digits database described in Subsection 7.1.2. We use this data to illustrate the abilities of FCMK and RFCMK to learn local resolution weights and cluster real and high dimensional data. We use the same experimental setting as in Subsection 7.2.2. That is, we use ten clusters for all algorithms and fix the fuzzifier m to 1.7 for fuzzy algorithms. For Spectral and DBSCAN, we tune the neighborhood parameter from 1 to 20 by an increment of 1. For kNERF, we tune the scaling parameter

Table 7.4: Spectral clustering results on the COREL data

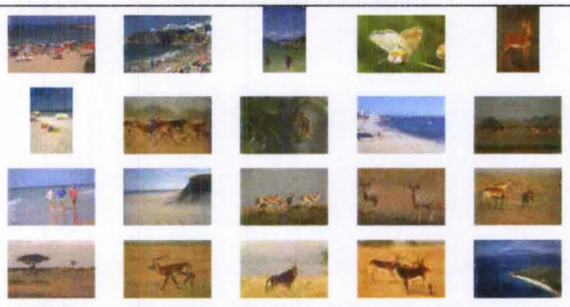
Cluster	Representative image	# images	Top 20 representative images
1		113	
2		98	
3		134	
4		76	
5		79	

Table 7.5: RFCMK clustering results on the COREL data

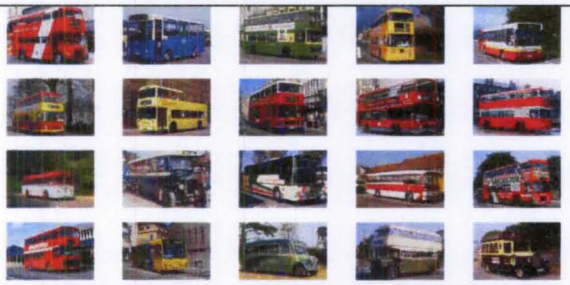
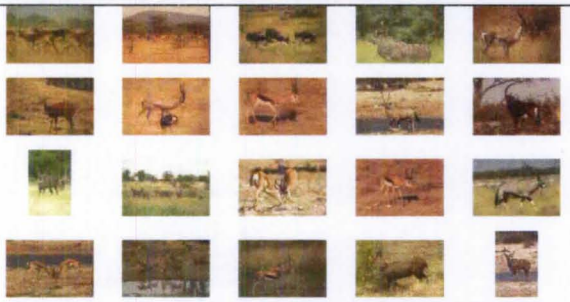
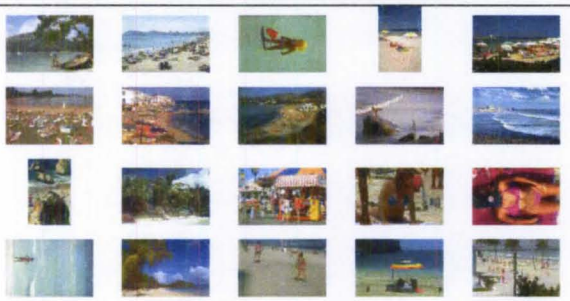
Cluster	Representative image	# images	Top 20 representative images
1		92	
2		101	
3		106	
4		92	
5		109	

Table 7.6: Resolution weights learned by RFCMK for the COREL dataset

Cluster	$\sigma_k = 0.1$	$\sigma_k = 0.15$	$\sigma_k = 0.2$	$\sigma_k = 0.5$	True σ
1	0.097	0.286	0.466	0.151	0.190
2	0.840	0.160	0	0	0.078
3	0	0.042	0.740	0.218	0.208
4	0.125	0.312	0.457	0.106	0.182
5	0.28	0.39	0.33	0	0.151

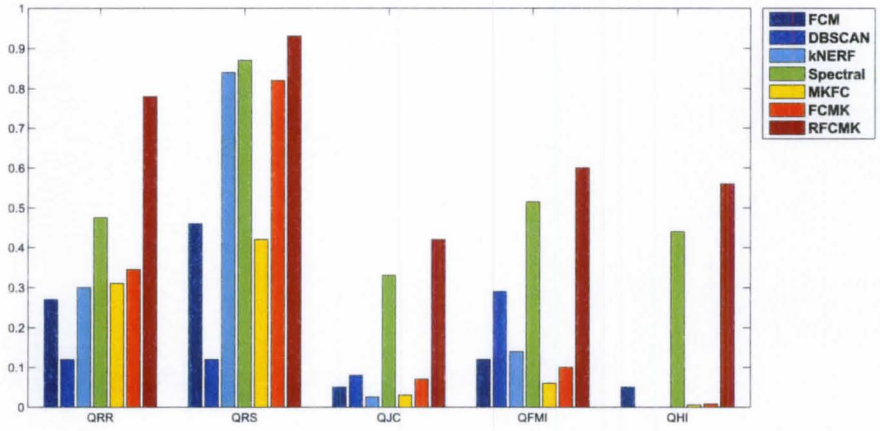


Figure 7.9: Performance measures obtained on categorizing the Pen Digits dataset using FCM, DBSCAN, kNERF, Spectral, MKFC, FCMK, and RFCMK clustering approaches.

between 0.01 and 100 with a step of 0.1. Since we should not use the ground truth to select the optimal parameter, for each algorithm, we select the parameter that gives the partition with the minimum sum of intra-cluster dissimilarities.

We compare our clustering results to those obtained using FCM [15], DBSCAN [70], kNERF [88], Spectral [118], and MKFC [57], using the measures described in Subsection 7.1.3.

Figure (7.9) compares the performance of the considered algorithms. As previously reported on other high dimensional data, prototype based clustering (FCM and FCMK) are not able to categorize this data correctly. Similarly, spectral and DBSCAN were not able to provide a good categorization of this data. This may be due to the selected parameters. As it can be seen in Figure (7.9), RFCMK outperforms the other methods. Since the spectral algorithm has the closest performance measure to RFCMK, we will use the partitions of these algorithms to compare and analyze the results in more details.

Table 7.7: Spectral clustering results on the Pen Digits dataset

		True Class										
		"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	Total
Clusters	1	0	8	0	4	0	0	0	51	0	3	66
	2	14	34	0	0	0	0	0	0	0	0	48
	3	0	0	0	0	102	0	0	0	0	7	109
	4	0	0	0	0	0	0	108	0	0	0	108
	5	63	0	0	1	0	57	0	0	109	70	300
	6	0	51	49	0	0	0	0	1	0	0	101
	7	0	0	0	115	0	3	0	0	0	4	122
	8	0	5	58	0	0	0	0	41	0	0	104
	9	1	1	0	1	6	59	6	14	0	27	115
	10	50	32	0	1	0	0	0	10	0	0	93

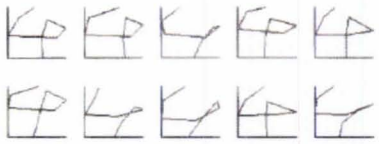
The spectral algorithm performed relatively better than the other state of the art approaches. However, when we analyze the partition obtained by the spectral algorithm in Table 7.7, we notice that it created a big cluster of 300 elements that contains a mixture of categories "0", "5", "8" and "9". We can also notice that category "1" was split in three clusters (cluster 2, 6 and 10). However, category "1" was not well segregated in clusters 6 and 10. This may be due to the fact that adapting one global scaling parameter to each sample resulted in distorting the structure of the data.

On the other hand, RFCMK was able to partition the data better than spectral. As it can be seen in Table 7.8, some digits have been categorized using more than one cluster (digits "0", "4", "5" and "8"). This is due to the different writing styles. For instance, digit "4" has been categorized using two clusters (cluster 6 and 8). Figure (7.10) displays the top 10 representatives of these two clusters (corresponding to the highest memberships). We can notice that each cluster contains one style of writing the digit "4". On the other hand, RFCMK categorized the other digits using mostly one cluster (digits "1", "2", "3", "6", "7" and "9"). These are categories with more consistent writing style. Figure (7.11) displays the top 10 representatives of cluster 3 (digit "3").

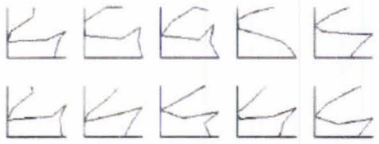
Table 7.9 reports the resolution weights learned by RFCMK for the Pen Digits dataset. For instance, for cluster 1, which includes mainly digit "2", the highest weight was assigned to $\sigma_k = 84.4$. Similarly, for cluster 10 which includes mainly digit "0", the highest weight

Table 7.8: RFCMK clustering results on the Pen Digits dataset

		True Class										
		"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	Total
Clusters	1	0	25	89	0	0	0	0	1	2	0	117
	2	0	0	0	0	0	0	105	1	0	0	106
	3	0	1	0	116	0	34	0	12	0	14	177
	4	0	2	18	0	0	0	0	85	0	0	105
	5	0	84	0	5	0	0	0	18	0	8	115
	6	45	0	0	0	48	0	0	0	0	3	96
	7	45	0	0	1	0	2	0	0	0	76	124
	8	0	0	0	0	45	57	0	0	41	8	151
	9	0	0	0	0	0	0	0	0	64	1	65
	10	38	19	0	0	15	26	9	0	2	1	110



(a)



(b)

Figure 7.10: Top 10 representatives of (a) cluster 6 and (b) cluster 8 obtained by RFCMK

Figure 7.11: Top 10 representatives of cluster 3 obtained by RFCMK

Table 7.9: Resolution weights learned by RFCMK for the Pen Digits dataset

Cluster	$\sigma_k = 94.9$	$\sigma_k = 84.4$	$\sigma_k = 73.7$	$\sigma_k = 60.9$	$\sigma_k = 44.7$	$\sigma_k = 25$
1	0.179	0.213	0.141	0.131	0.117	0.113
2	0.189	0.140	0.231	0.129	0.110	0.105
3	0.159	0.150	0.142	0.139	0.137	0.137
4	0.183	0.219	0.141	0.130	0.115	0.110
5	0.175	0.204	0.141	0.133	0.120	0.117
6	0.225	0.186	0.140	0.129	0.113	0.108
7	0.159	0.141	0.177	0.137	0.131	0.129
8	0.232	0.189	0.140	0.128	0.110	0.105
9	0.176	0.141	0.206	0.133	0.119	0.115
10	0.221	0.184	0.141	0.130	0.114	0.109

Table 7.10: Simulation time (in seconds) for the Spectral and the RFCMK algorithms

	# instances	# features	# clusters	Spectral	RFCMK
dataset 1	150	2	3	1.49 s	1.31 s
dataset 2	490	2	3	26.28 s	14.08 s
dataset 3	400	2	3	20.93 s	11.32s
dataset 4	1016	2	2	35.58 s	14.71 s
dataset 5	393	2	3	21.03 s	11.27 s
dataset 6	547	2	3	9.90 s	6.89 s
dataset 7	200	2	2	3.81 s	2.61 s
COREL data	500	182	5	49.68 s	16 s
Pen Digits data	1166	16	10	1323.05 s	144.20 s

was assigned to $\sigma_k = 94.9$. Despite the different writing styles, RFCMK categorized most of the digits using only one cluster.

7.2.4 Time complexity

Since the spectral algorithm has the closest performance measure to RFCMK, we compare the RFCMK and the spectral [118] methods by computing their computational time. As we can notice from Table 7.10, RFCMK is computationally less expensive than spectral. This becomes more significant as the size of the dataset increases. In fact, though spectral

can produce high quality clustering on small datasets, it has limited applicability to large scale problems due to its computational complexity of $\mathcal{O}(N^3)$ where N is the number of data points [58].

7.2.5 Conclusions

The experimental results presented in this section demonstrate the effectiveness of the proposed clustering algorithms. The incorporation of multiple kernels and unsupervised adjustment of the kernel weights within each cluster makes the choice of the kernels less crucial and allows better characterization and adaptability to each individual cluster. This is very helpful when we deal with real and high dimensional data where the range of possible values for the parameter to be tuned may not be evident. For instance, in the case of 2D experiments, we have selected the neighborhood parameter for spectral based on visualizing the results and selecting the best one. Although this can be performed on 2D dataset, it is not possible for high dimensional dataset where visualization may not be trivial and the ground truth is not known.

7.3 Evaluation of the semi-supervised algorithms

In this section, we compare the semi-supervised versions of FCMK and RFCMK (SS-FCMK and SS-RFCMK) clustering results to those obtained by the most related semi-supervised clustering approaches. Namely, we compare our results to those obtained using the semi-supervised Fuzzy C-Means [78], the semi-supervised kernel C-means (SS-kernel-C-Means) [7], and the semi-supervised Spectral algorithm [29]. To assess the performance of the different clustering algorithms and compare them, we use the ground truth, and the same five performance measures described in Subsection 7.1.3.

Ideally, the supervision information should be provided by the user (in terms of feedback) in an interactive mode. However, to carry out an objective experiment (for the purpose of algorithm evaluation), we automate the process of constraints selection and attempt to simulate the user's feedback. We assume that boundary points are the most informative points, and that these points should be selected for the supervision information. First, we ran the considered algorithms without supervision for few iterations. Then, we identify 2% of boundary points (points with low fuzzy membership values in

all clusters). Next, we use the ground truth of these points to construct must-link and cannot-link constraints.

7.3.1 Synthetic datasets

We have shown in Subsection 5.1.2 and Subsection 5.2.2 the enhancement of the clustering results of FCMK and RFCMK on 2D datasets by the use of a small amount of side information. In order, to better assess the performance of SS-FCMK and SS-RFCMK on these datasets, we compare their clustering results with other clustering approaches as mentioned above. We use the four synthetic datasets where the unsupervised algorithms (FCMK and RFCMK) did not perform well. These 4 datasets are displayed in Figures 3.2(b), 3.2(c), 3.2(d) and 4.1(b).

Figure (7.12) displays the clustering results on dataset 2. As it can be seen, neither SS-kernel-C-Means (Figure 7.12(b)) or SS-Spectral (Figure 7.12(c)) were able to categorize this data correctly. This is because these algorithms use a global scaling parameter. Thus, they are not able to deal with the local characteristics of this dataset even when partial supervision is provided. On the other hand, the overlapping boundaries between the three clusters are better characterized using SS-FCMK and SS-RFCMK. Thus, the partial supervision guided effectively the FCMK and RFCMK algorithms towards a better categorization of the data.

Figure (7.13) displays the clustering results on dataset 3. As it can be seen in Figure, SS-FCM was not able to partition this data correctly. This is mainly because SS-FCM is designed to seek compact spherical clusters. The SS-kernel-C-Means was not able to categorize this data. Similar to previous example, this is due to the one global scaling parameter used by this algorithm. On the other hand, the partial supervision has guided FCMK to a better partition of the dataset (Figure 7.13(d)).

Figure (7.14) displays the clustering results on dataset 4. The SS-FCM seeks spherical clusters and thus cannot categorize this data correctly as shown in Figure 7.14(a). We can notice that regardless of the multi-resolution of the different clusters in the original feature space, SS-kernel-C-Means approach maps all the points using the same global distance measure. This is not appropriate for this data as it can be seen in Figure 7.14(b) even when partial supervision is provided. On the other hand, the partial supervision has guided SS-Spectral, SS-FCMK and SS-RFCMK to partition this data correctly.

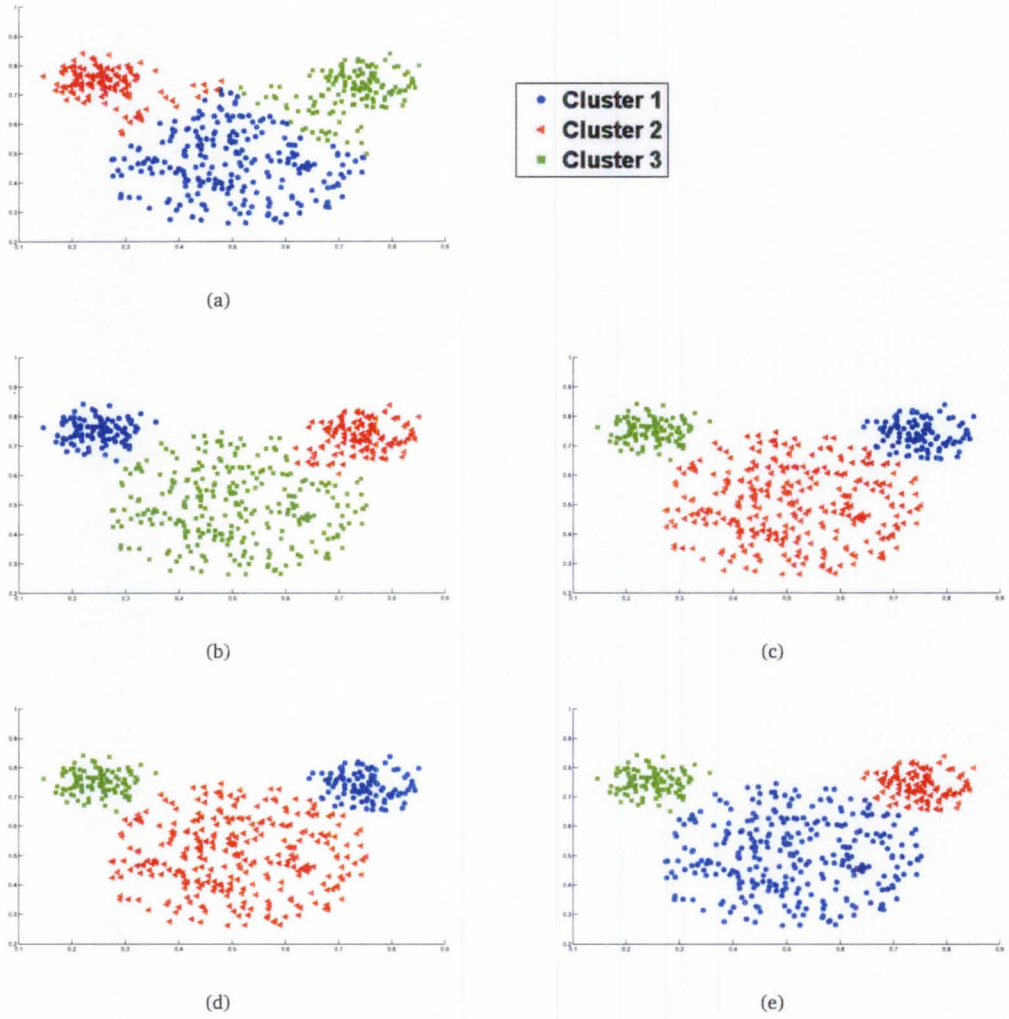


Figure 7.12: Results of clustering dataset 2 using (a) SS-FCM, (b) SS-kernel-C-Means, (c) SS-Spectral, (d) SS-FCMK, and (e) SS-RFCMK.

Figure (7.15) displays the clustering results on dataset 6. As with previous data sets, The SS-FCM, SS-kernel-C-Means and SS-FCMK were not able to partition this data correctly. On the other hand, the partial supervision has guided SS-RFCMK to partition this data correctly.

7.3.2 Application to image database categorization

To evaluate the performance of SS-FCMK and SS-RFCMK on the COREL dataset, we compare their results to those obtained using the semi-supervised Fuzzy C-Means [78], the semi-supervised kernel C-means (SS-kernel-C-Means) [7], and the Semi-Supervised Spectral Learning algorithm [29]. We first perform the unsupervised versions of the five

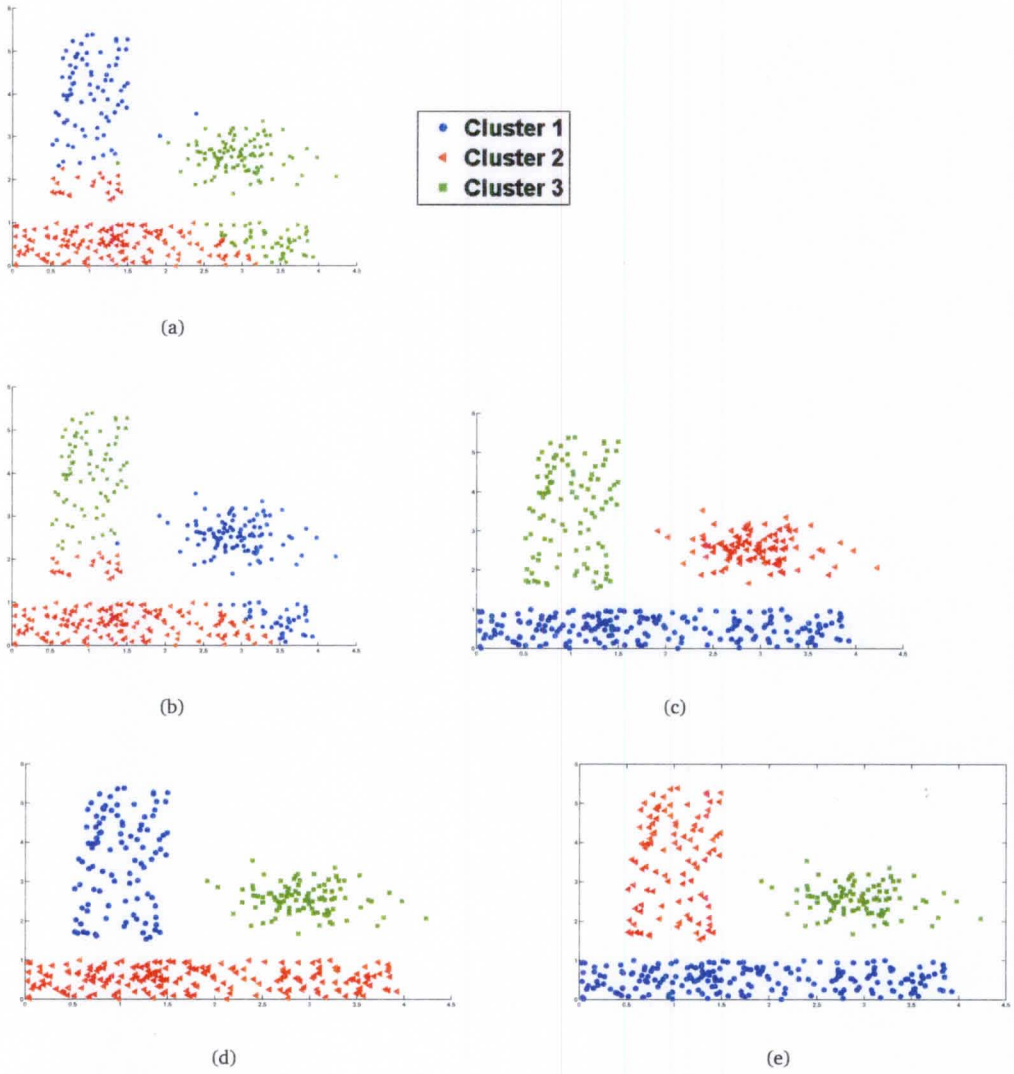


Figure 7.13: Results of clustering dataset 3 using (a) SS-FCM, (b) SS-kernel-C-Means, (c) SS-Spectral, (d) SS-FCMK, and (e) SS-RFCMK.

considered algorithms. Then, we run their corresponding semi-supervised versions 10 times and incrementally add 1% of the total number of images as pairwise constraints each time. The performance of the five algorithms, measured in terms of classification rate, on the multiple runs is shown in Figure (7.16). As it can be seen, the performance of all the algorithms is enhanced when supervision increases. However, the SS-RFCMK has the best performance for this dataset. On the other hand, a small amount of supervision has allowed SS-FCMK to improve significantly its performance as it can be seen in Figure (7.16). For instance, when the percentage of data samples used for supervision is equal

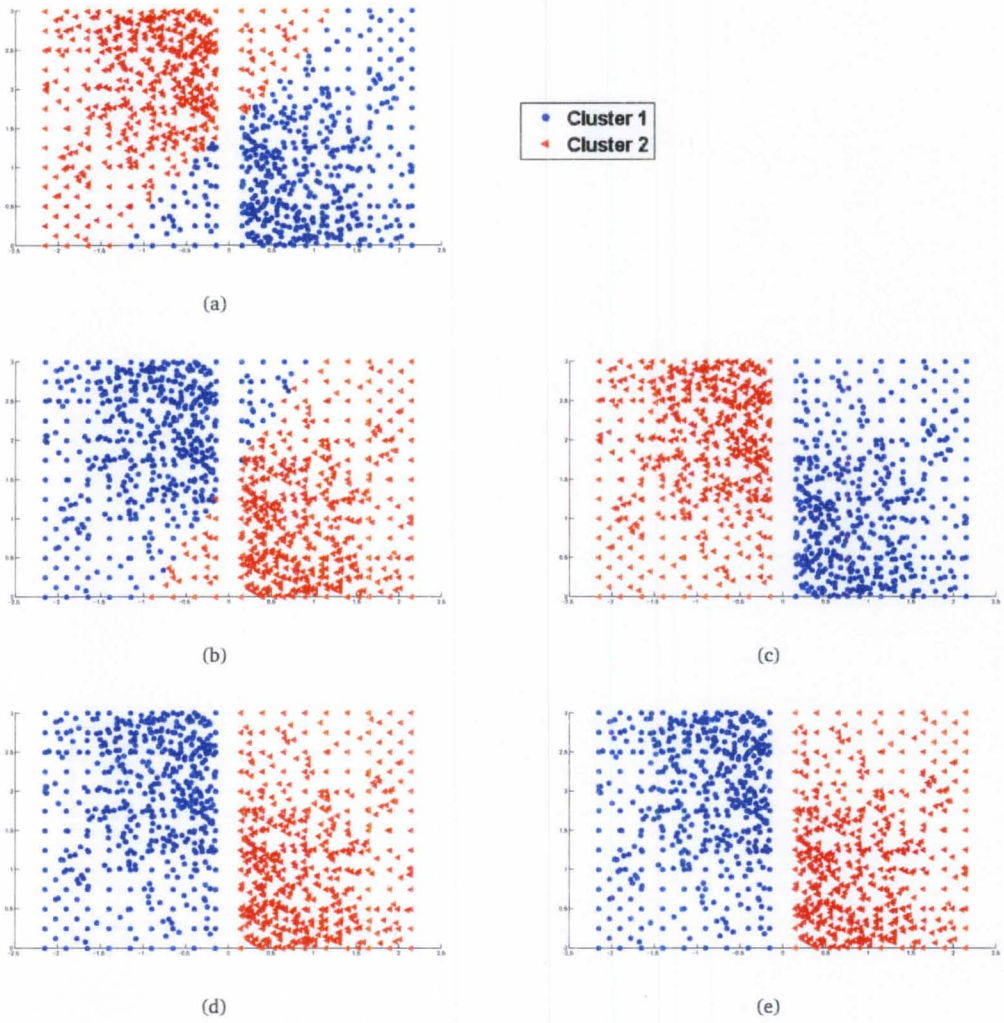


Figure 7.14: Results of clustering dataset 4 using (a) SS-FCM, (b) SS-kernel-C-Means, (c) SS-Spectral, (d) SS-FCMK, and (e) SS-RFCMK.

to 10%, SS-FCMK outperforms SS-Spectral and has the same performance as SS-kernel-C-Means.

Table 7.11 compares the number of images used to construct the constraints and the number of images with improved categorization when the constraint ratio is equal to 2%. We note that the number of images with improved categorization is much higher than the number of images used to construct the constraints. That is, the 2% of supervision information has guided the algorithm to categorize more images correctly.

As mentioned earlier, the partial supervision information is used to guide the clustering algorithm and can make them less sensitive to initialization. To illustrate this point, we compare the performance of the unsupervised and semi-supervised algorithm (with

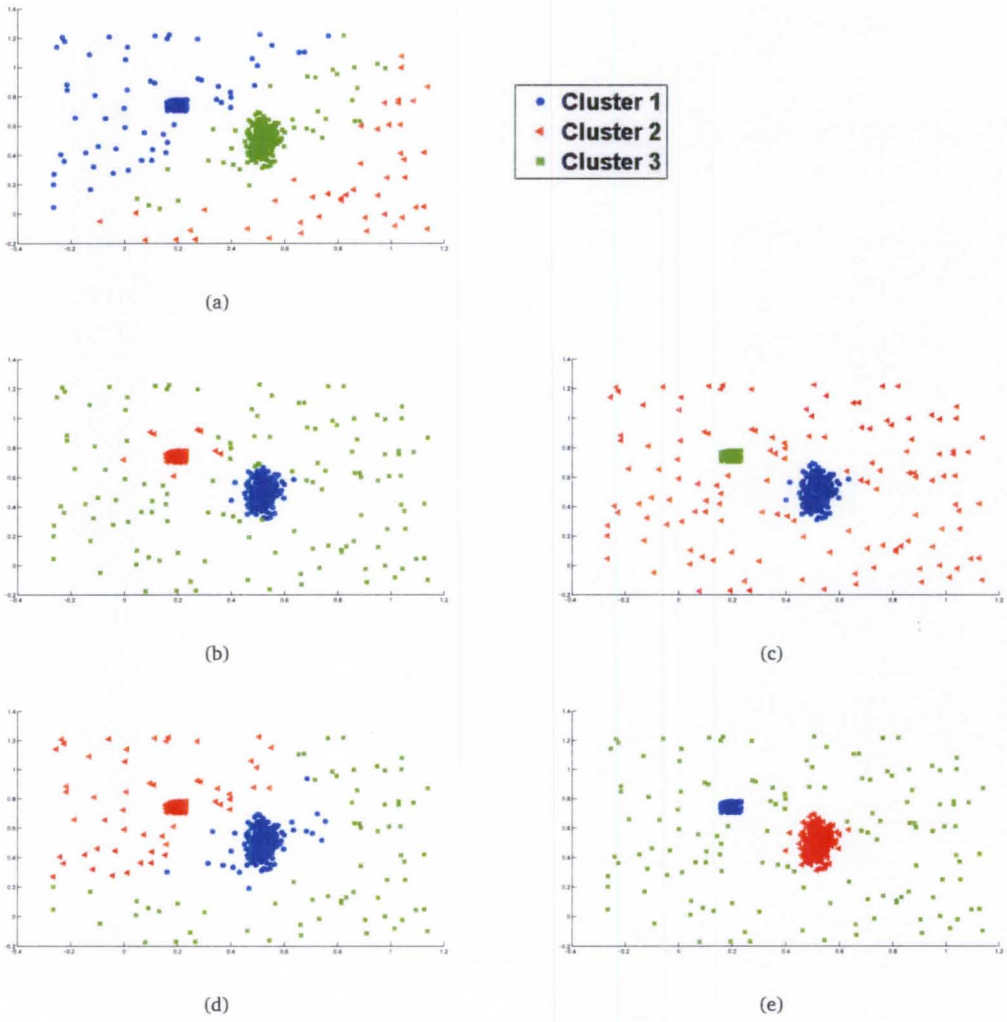


Figure 7.15: Results of clustering dataset 6 using (a) SS-FCM, (b) SS-kernel-C-Means, (c) SS-Spectral, (d) SS-FCMK, and (e) SS-RFCMK.

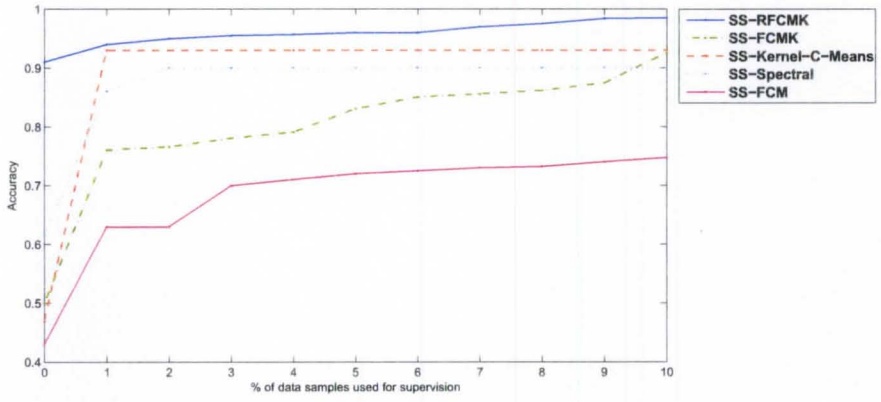


Figure 7.16: The accuracy of the five semi-supervised algorithms on COREL data vs. the number of pairwise constraints.

Table 7.11: Number of images used to construct the constraints vs. the number of images with improved categorization

Algorithm	Number of images used to construct the constraints	Number of images with improved categorization
SS-FCMK	10	57
SS-RFCMK	10	19

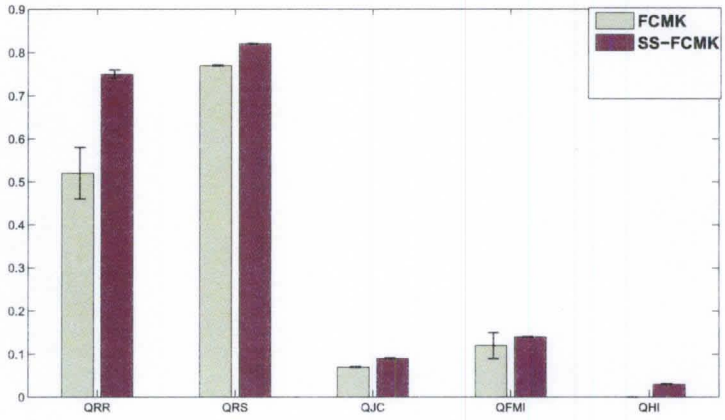


Figure 7.17: Mean and standard deviation of the five performance measures over 20 runs of FCMK and SS-FCMK on the COREL dataset.

2% of supervision) for 20 runs with different initializations. We use the performance measures defined in Subsection 7.1.3.

Figure (7.17) compares the mean and standard deviation of the five performance measures obtained on the COREL dataset using FCMK and SS-FCMK. Similarly, Figure (7.18) compares the mean and standard deviation of the performance measures using RFCMK and SS-RFCMK. First, we note that the semi-supervised algorithms outperform the unsupervised version since their performance mean is always larger. Second, the standard deviation of the performance of the semi-supervised algorithm is much smaller than that of the unsupervised version. This indicates that the partial supervision makes the algorithm less sensitive to initialization.

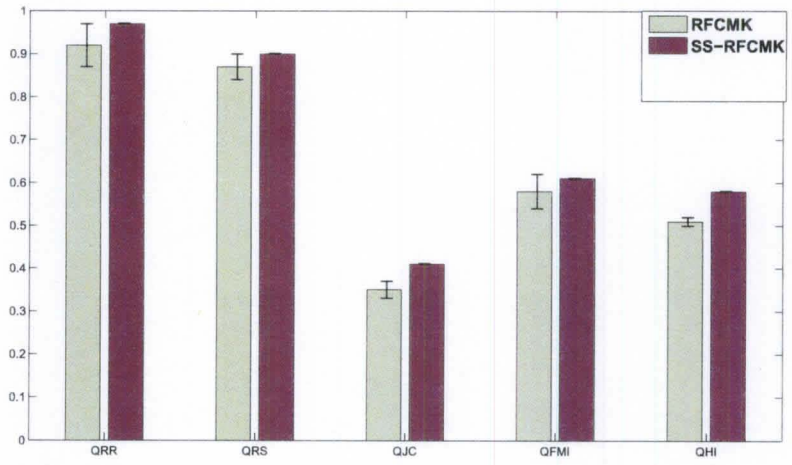


Figure 7.18: Mean and standard deviation of the five performance measures over 20 runs of RFCMK and SS-RFCMK on the COREL dataset.

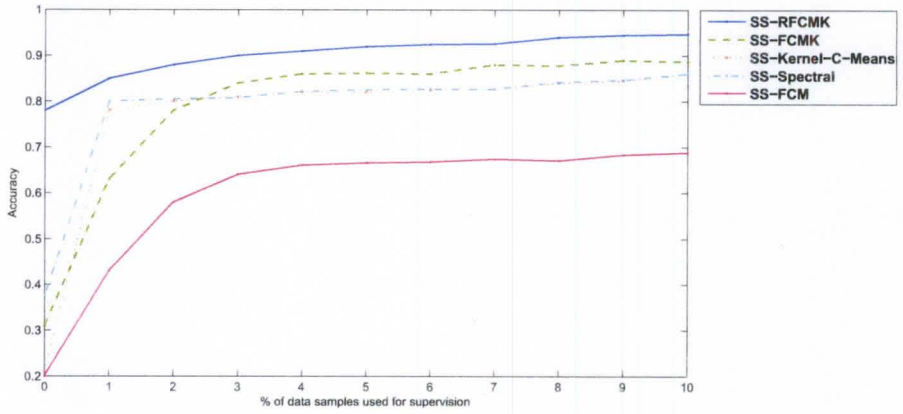


Figure 7.19: The accuracy of the five semi-supervised algorithms on Pen Digits data vs. the number of pairwise constraints.

7.3.3 Application to categorization of handwritten digits

Figure (7.19) reports the accuracy of the five considered algorithms as we increase the amount of supervision information. As expected, the accuracy of all algorithms improve as we increase the number of constraints. Moreover, as it can be seen, SS-FCMK outperforms SS-kernel-C-Means and SS-Spectral when the percentage of data sample used for supervision is above 2%. We also note that SS-RFCMK outperforms all the other approaches.

Table 7.12: Number of digits used to construct the constraints vs. the number of digits with improved categorization

Algorithm	Number of digits used to construct the constraints	Number of digits with improved categorization
SS-FCMK	23	105
SS-RFCMK	23	56

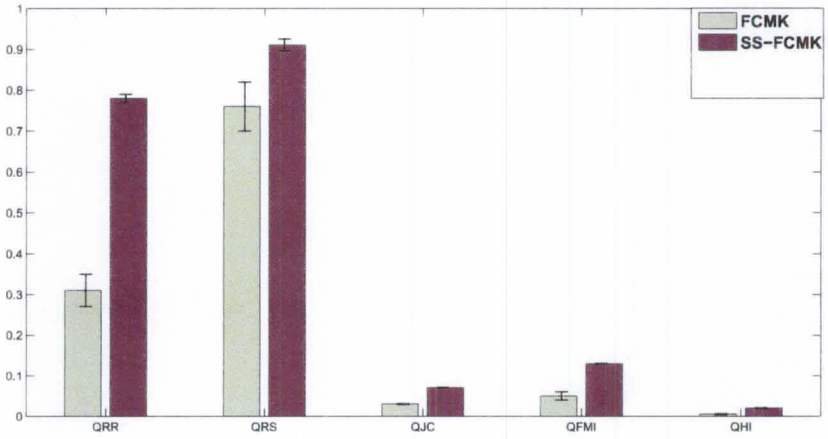


Figure 7.20: Mean and standard deviation of the five performance measures over 20 runs of FCMK and SS-FCMK on the Pen Digits dataset.

Table 7.12 reports the number of digits used to construct the constraints versus the number of digits with improved categorization when the ratio of constraints is set to 2%. As it can be seen, the number of digits with improved categorization is much larger than the number of digits used to construct the constraints. This indicates that the partial supervision has guided the algorithm to a better optima.

Figure (7.20) and Figure (7.21) report the mean and standard deviation of the performance measures of FCMK versus SS-FCMK and RFCMK versus SS-RFCMK, respectively, over 20 runs with different initializations. As with the image database, we note that in addition to giving better performance measures than their respective unsupervised clustering versions, the SS-FCMK and SS-RFCMK are less sensitive to initialization and their final partition is more consistent.

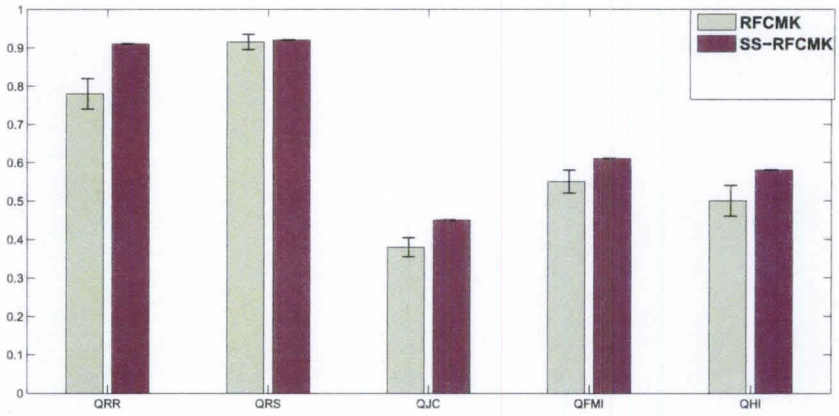


Figure 7.21: Mean and standard deviation of the five performance measures over 20 runs of RFCMK and SS-RFCMK on the Pen Digits dataset.

7.3.4 Conclusions

In this section, we have illustrated the clustering performance of SS-FCMK and SS-RFCMK on both synthetic 2D datasets and real image datasets. We have shown that SS-FCMK and SS-RFCMK outperform other related semi-supervised clustering algorithms on 2D synthetic data. In particular, we have shown that a small subset of constraints can guide the algorithm towards a more optimal partition, and thus, improving the categorization of many more samples. We have also shown that the partial supervision can make the algorithm less sensitive to initialization and local minima.

7.4 Evaluation of the RFCMK algorithm for VL data

We compare the performance of the VL RFCMK algorithms on the following datasets. The synthetic 2D50 dataset displayed in Figure (7.22) is composed of 7,500 instances with 2 numeric attributes and 50 classes. The MNIST dataset is a subset of the collection of handwritten digits available from the *National Institute of Standards and Technology* (NIST). The MNIST dataset includes 70,000 28x28 pixel images of the digits 0 to 9. each pixel has an integer value between 0 to 255. We normalize the pixel values to the interval $[0, 1]$ by dividing by 255 and concatenate each image into a 784-dimensional vector.
To assess the performance of the VL RFCMK algorithms, we, first, compare their clustering results to that of RFCMK applied on the full dataset. Second, we show that these algorithms can produce reasonable partitions of large datasets. Each experiment was

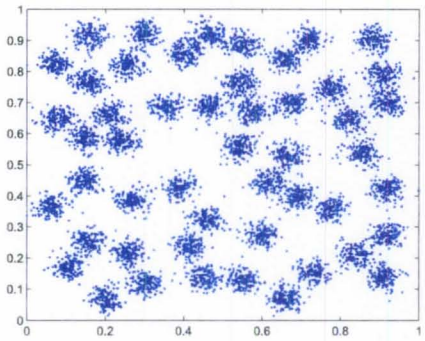
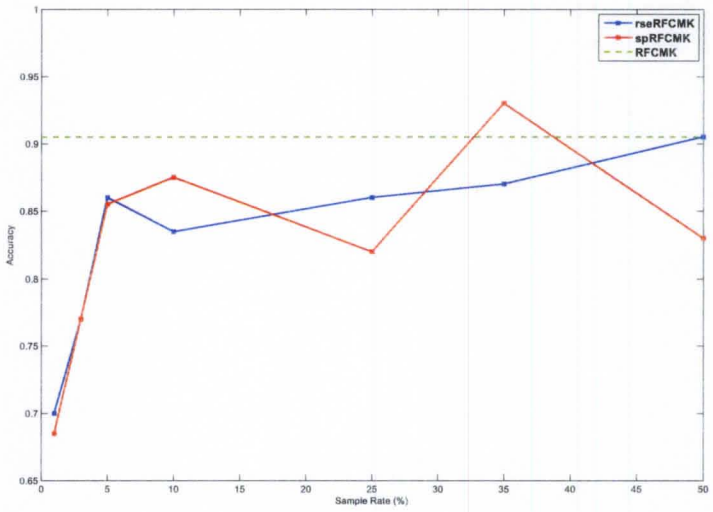


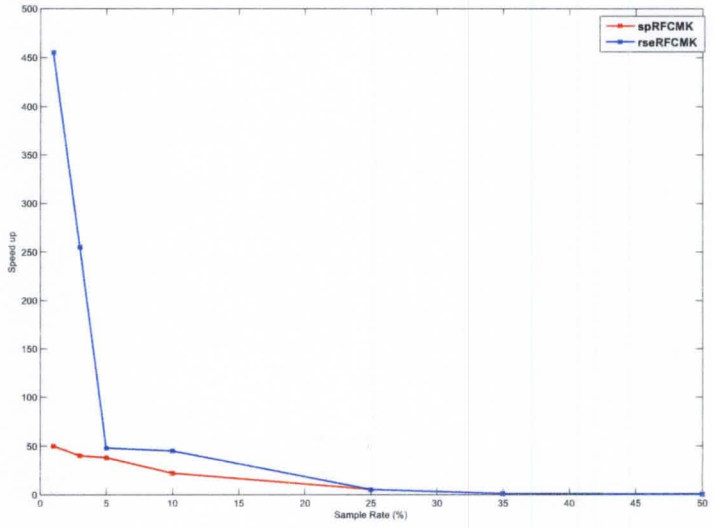
Figure 7.22: 2D50 synthetic data; $N = 7,500$ objects, $C = 50$ clusters

conducted with 50 random initializations. Each data set was clustered using RFCMK, loading all the data into memory, and VL RFCMK, loading a chosen percentage of data into memory. For measuring quality, we compute the accuracy relative to ground truth labels. We also compare the speed-up obtained by rseRFCMK and spRFCMK compared with RFCMK, when loading the entire data set into memory before clustering. The experiments were performed on a dedicated single core of a server with 96GB of memory. All code was written in the MATLAB computing environment.

Figure (7.23) shows the results of the VL RFCMK algorithms on the 2D50 data set. As it can be seen in Figure 7.23(a), the rseRFCMK algorithm shows slightly better results than spRFCMK at low sample rates ($< 5\%$). With 10% data loaded, spRFCMK outperforms rseRFCMK, with average quality almost the same as RFCMK. However, for high sample rates ($> 10\%$), the spRFCMK algorithm exhibits inconsistent performance. This is maybe due to the fact that the number of clusters C is fixed. Figure 7.23(b) indicates the average speed-up of VL RFCMK compared to RFCMK on the 2D50 data set. As it can be seen, loading and clustering 1% of the data is faster than loading 10% of the data for both approaches. We notice that rseRFCMK is faster than spRFCMK at low sample rates. For instance, rseRFCMK achieved an excellent speed-up (above 450 times) at 1% sample rate, while spRFCMK speedup is about 50 times. However, at high sample rates ($>$ 10%), the two algorithms perform comparably with nearly equal speed-up factors. We should mention that RFCMK loads all data in memory, so the speed-up reported here for spRFCMK is the minimum. For large data sets that cannot fit in memory, RFCMK will require multiple disk accesses making it slower. The results indicate that spRFCMK can be used for speeding up RFCMK even if the data can be fully loaded into memory.

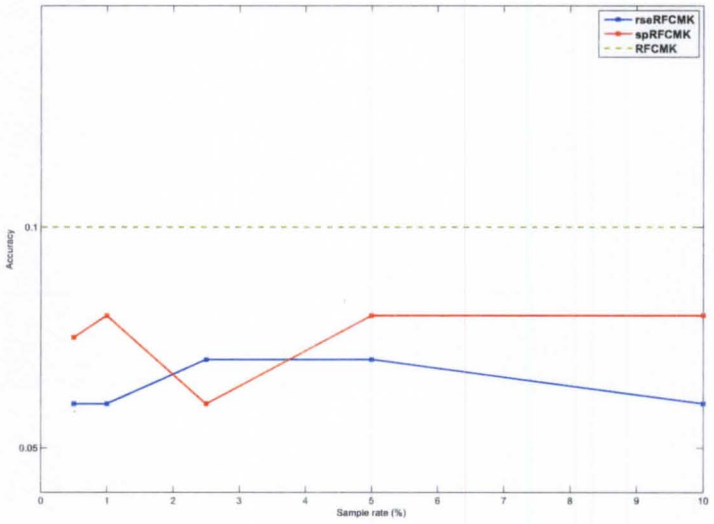


(a)

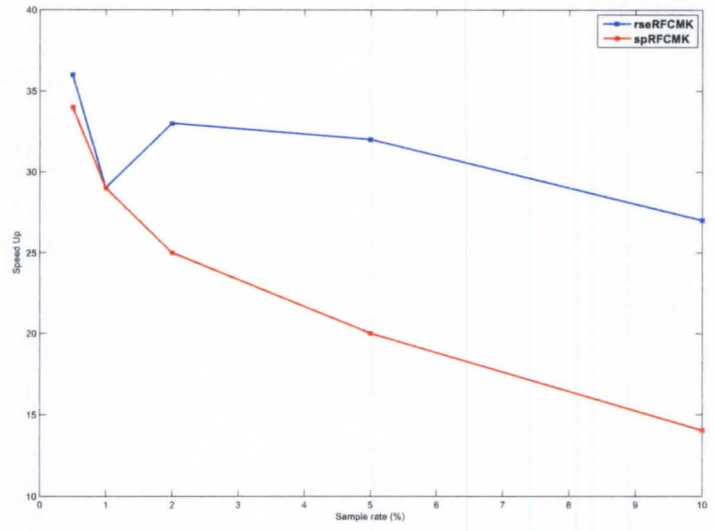


(b)

Figure 7.23: Performance of rseRFCMK and spRFCMK algorithms on 2D50 dataset in terms of (a) Accuracy and (b) Speed Up.



(a)



(b)

Figure 7.24: Performance of rseRFCMK and spRFCMK algorithms on MNIST dataset in terms of (a) Accuracy and (b) Speed Up.

Figure 7.23(b) indicates the average speed-up of VL RFCMK compared to RFCMK on the MNIST data set. At low sample rates, the spRFCMK and rseRFCMK perform comparably. This is because rseRFCMK suffered from slow convergence at the low sample rates. At sample rates >2%, rseRFCMK is clearly the fastest algorithm. However, the accuracy results in Figure 7.24(a) show a dismal view of the performance of these algorithms for this dataset. Both algorithms show markedly inferior performance to RFCMK. We should note that even RFCMK did not perform well on this dataset in terms of its accuracy with respect to comparison with the ground truth labels. We believe that this is because the MNIST data do not cluster well relative to its 0-9 ground truth labels. Hence, the cluster structure is degraded by sampling aspect of the spRFCMK and rseRFCMK algorithms.

In this section, we have illustrated the performance of VL RFCMK based on speed up and quality of approximations to batch RFCMK on both synthetic 2D dataset and real image dataset. We have shown that VL RFCMK are able to achieve good quality partitions, with an average quality almost the same as RFCMK, by loading less than 10% of the data. Moreover, our VL RFCMK algorithms provide significant speed up even when compared to clustering all the data at once in memory. So, besides using them for clustering large data sets they can also be used for speed up purposes even for data that can be fully loaded into memory. The rseRFCMK algorithm seems to be the preferred algorithm. It is the most scalable and efficient solution, and produces results comparable to those of RFCMK.

CHAPTER 8

CONCLUSIONS AND FUTURE WORKS

Clustering is useful in several exploratory pattern analysis, grouping, decision making and machine learning situations including data mining, document retrieval, image segmentation and pattern classification. However, clustering is a difficult problem combinatorially, especially when the structure of the data does not correspond to easily separable categories with unbalanced sizes, densities and shapes. In order to make the problem easier, clustering was extended to kernel based approaches that allow the use of a specific similarity measure. The choice of kernel function allows the mapping of the input data into a new space in such a way that computing a simple partitioning in this feature space results in a nonlinear partitioning in the input space. The Gaussian kernel is one of the most used kernel functions due to its analytical proprieties. However, its performance depends on the selection of the scaling parameter σ among an extensive range of possibilities. The problem is aggravated for many real-world clustering applications, when there are large variations between the distributions of the different clusters in the feature space. Existing kernel based approaches using one global parameter for the entire data set ignore the possible disparity of the data structure in different spatial areas.

One way to learn optimal scaling parameters is to try several combinations, evaluate each partition using some validity measure, and identify the optimal partition. However, this exhaustive search of one scaling parameter with respect to each cluster is not practical. It is computationally expensive and increases significantly with the number of clusters

and the range of possible values of σ_i. Moreover, it may not be possible to quantify the optimal partition.

In this dissertation, we addressed the above limitations and built multiple kernel algorithms which make a better tradeoff between flexibility and tractability. Here, we proposed two novel fuzzy clustering techniques that learn multiple kernel similarity measure uniform over clusters in an efficient way: the Fuzzy C-means with Multiple Kernels algorithm (FCMK) and the Relational Fuzzy Clustering with Multiple Kernels approach (RFCMK).

The Fuzzy C-Means with Multiple Kernel (FCMK) simultaneously finds the best degrees of membership and the optimal cluster-dependent kernel weights for a combination of a set of multi-resolution Gaussian kernels. Our main observation is that data from the same cluster tend to manifest similar properties, thus the intra-cluster weights can be approximately uniform while kernel weights are allowed to vary over clusters. FCMK minimizes one objective function for both the optimal partition and the resolution weights for each Gaussian kernel in each cluster. This optimization is done iteratively by dynamically updating the partition, the prototypes, and the resolution weights in each iteration. This makes the proposed prototype-based algorithm simple and fast. The incorporation of multiple kernels and the automatic adjustment of kernel weights render FCMK more immune to unreliable kernels. FCMK also makes combining kernels more practical since appropriate weights are assigned automatically. Effective kernels tend to contribute more to the clustering and therefore improve results. However, since it is based on Kernelized metric FCM [124], FCMK inherits the limitations of object-based clustering. In particular, multiple runs of the algorithm may be needed since FCMK is susceptible to initialization problems. Moreover, FCMK is not suitable for all types of data. In FCMK, the density is estimated by counting the number of points within a specified radius, σ_k, of the center. However, we seek to estimate the variance between pairs of points and not point-to center to detect the multiple resolutions within the same cluster. Finally, FCMK is restricted to data for which there is a notion of center (centroid). In some applications, the set of objects may be represented by relational data.

The Relational Fuzzy C-Means with Multiple Kernel (RFCMK) is another algorithm that we have developed that learns convex combination of kernel functions with cluster dependent kernel weights while seeking compact clusters. Compared to the first proposed

algorithm (FCMK), this algorithm is applicable when data cannot be represented by feature vectors and only the degree to which pairs of objects in the data are related is available. Moreover, even if the data can be represented by feature vectors, the RFCMK is more practical when similar objects cannot be represented efficiently by a single prototype. Similar to the FCMK, the RFCMK constructs the kernel from a number of multi-resolution Gaussian kernels and learns a resolution-specific weight for each kernel function in each cluster. This allows better characterization and adaptivity to each individual cluster. RFCMK minimizes one objective function for both the optimal partition and for the cluster dependent kernel weights. This optimization is done iteratively by dynamically updating the partition and the local measure in each iteration. For instance, the multiple kernel learning task takes advantage of the unlabeled data and reciprocally, the categorization task takes advantage of the local learned combination of kernels.

To the best of our knowledge, FCMK and RFCMK are the first fuzzy algorithms with a cluster-dependent multiple kernel learning setting in an unsupervised way. This is a major contribution to Gaussian-based clustering approaches such as kernel and spectral clustering methods that suffer from their sensitivity to the choice of the scaling parameter. We have illustrated the clustering performance of FCMK and RFCMK on synthetic 2D datasets and on high dimensional real data. Our experimental results on 2D datasets have demonstrated the effectiveness of FCMK and RFCMK. In addition, we showed that the learned resolution weights and the fuzzy memberships returned by FCMK and RFCMK are meaningful and reflect the geometric characteristic of the data. We have shown that the resolution weights leaned by RFCMK are not only influenced by the intra-cluster distances, but also by the relative cluster positions, densities and sizes. This allows a better description of the data and consequently, a better partition of the data. Our experiments on real and high dimensional data have indicated that FCMK may not perform well on high dimensional data. One reason for this suboptimal behavior is that FCMK is a prototype-based method where the density is estimated by counting the number of points within a specified radius, σ_k, of the center. RFCMK was designed to overcome these limitation and has proved to outperform other algorithms.

Both the FCMK and RFCMK algorithms minimize complex objective functions that are prone to local minima and sensitive to initialization. This problem is more acute for high dimensional datasets. Thus, if a small amount of prior knowledge is available, it can

be used to guide the clustering algorithms to avoid most local minima and obtain a better partition. In the second part of this dissertation, we presented two semi-supervised clustering approaches: the Semi-Supervised Fuzzy C-Means with Multiple Kernels (SS-FCMK), and the Semi-Supervised Relational Fuzzy Clustering with Multiple Kernels (SS-RFCMK). We assume that for both algorithms we have a set of pairwise "Must-Link" constraints (pairs of points that should belong to the same cluster) and a set of "Cannot-Link" constraints (pairs of points that should belong to different clusters). We have illustrated the clustering performance of SS-FCMK and SS-RFCMK on synthetic 2D datasets and on high dimensional real data. We have shown that SS-FCMK and SS-RFCMK outperform other related semi-supervised clustering algorithms. In particular, we have shown that using a small subset of constraints can guide the algorithm towards a more optimal partition, and thus, improving the categorization of many more samples. We have also shown that the partial supervision can make the algorithm less sensitive to initialization and local minima.

The main limitation of the RFCMK is the lack of scalability. Currently, the proposed approach requires all data to be clustered to be available in memory. This may not be possible for very large data collections. As an attempt to overcome this limitation, we proposed two relational fuzzy clustering with multiple kernel algorithms for large data: the random sample and extend RFCMK (rseRFCMK) and the single pass RFCMK (spRFCMK) approaches. The rseRFCMK method computes cluster prototypes from a smaller sample of randomly selected objects, and then extends the partition to the remainder of the data. While spRFCMK sequentially loads manageable sized chunks, clustering these chunks in a single pass, and then combining the results from each chunk. We have illustrated the performance of rseRFCMK and spRFCMK on loadable datasets in terms of speed and quality of approximation to batch RFCMK. Our experiments have indicated that we achieved good quality partitions, with an average quality almost the same as RFCMK by loading less than 10% of the data. Moreover, rseRFCMK and spRFCMK provide significant speed up even when compared to clustering all the data at once in memory. So, besides using them for clustering large data sets they can also be used for speed up purposes even for data that can be fully loaded into memory. The rseFCMK algorithm seems to be the preferred algorithm. It is the most scalable and efficient solution, and produces results comparable to those of spFCMK.

There are a number of interesting potential avenues for future research in multiple kernel methods for fuzzy clustering to improve the performance of our algorithms even further. In particular, we propose investigating the following tasks.

8.1 Finding the optimal number of clusters

We are interested in automatically identifying the optimal number of clusters and reducing the effect of noise and outliers. This will be especially important for deploying our proposed clustering algorithm RFCMK for practical applications. Since the proposed approach is based on relational FCM, it can inherit most of the advantages of FCM-type clustering algorithms. For instance, techniques that were used to extend the RFCM to find the optimal number of clusters [86], and to reduce the effect of noise and outliers [103] could be adapted to RFCMK.

8.2 Feature weighting

When clustering complex data, the relationship among the objects may be described by multiple (dis)similarity matrices. For instance, in image database categorization, we may have one similarity matrix that encodes color information, another matrix for texture information, and another one for structure information. Existing relational clustering algorithms can operate on only one similarity matrix at a time. Thus, one has to partition each matrix independently, or partition a single matrix that combines all matrices in a uniform way. However, the influence of the features is generally not equally important in the definition of the category to which similar patterns belong. Therefore, to obtain meaningful clusters across all similarity matrices, we need to perform clustering and feature discrimination simultaneously. This can be achieved by integrating feature weighting in the process of Multiple Kernel Learning. The resulting algorithm would partition the data into clusters, and learns a resolution weight simultaneously. The learned resolution weight would be feature dependent with respect to each cluster.

REFERENCES

[1] S. Basu A. Banerjee and R. Mooney. Semi-supervised clustering by seeding. *Proc. Int. Conf. on Machine Learning*, pages 19–26, 2002.

[2] S. Basu A. Banerjee and R. Mooney. Active semi-supervision for pairwise constrained clustering. *Proc. the SIAM Int. Conf. on Data Mining*, pages 333–344, 2004.

[3] K Bennett A. Demiriz and M. Embrechts. Semi-supervised clustering using genetic algorithms. *IESTANN*, pages 809–814, 1999.

[4] P. Indyk A. Gionis and R. Motwani. Similarity search in high dimen- sions via hashing. *Proceedings of the 25th International Conference on Very Large Data Bases*, page 518Ð529, 1999.

[5] Y. S. Abu-Mostafa. *Machines that learn from hints*. Scientific American Inc, 1995.

[6] J.C. Bezdek A.M. Bensaid, L.O. Hall and L.P. Clarke. Partially supervised clustering for image segmentation. *Pattern Recognition*, 29:859–872, 1996.

[7] I. Dhillon B. Kulis, S. Basu and R. Mooney. Semi-supervised graph clustering: a kernel approach. *ICML*, 2005.

[8] R. Herbrich B. Scholkopf and A.J. Smola. A generalized representer theorem. *COLT/EuroCOLT*, pages 416–426, 2001.

[9] J. Kwok B. Zhao and C. Zhang. Multiple kernel clustering. *Proceedings of The 9th SIAM International Conference on Data Mining*, page 638Ð649, 2009.

[10] G.L.F. Bach and M. Jordan. Multiple kernel learning, conic duality and the smo algorithms. *ICML*, 2004.

[11] N. Baili and H. Frigui. Fuzzy clustering with multiple kernels. *IEEE Fuzzy Systems*, 2011.

[12] N. Baili and H. Frigui. Relational fuzzy clustering with multiple kernels. *ICDM*, 2011.

[13] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[14] A. Bensaid and J.C. Bezdek. Semi-supervised point prototype clustering. *International Joint Conference on Pattern Recognition and ArtiÞcial Intelligence*, 12:625–643, 1998.

[15] J. Bezdek. *Pattern Recognition with fuzzy objective function algorithm*. Plenum Press, New York, 1981.

[16] J.C. Bezdek and R.J. Hathaway. Progressive sampling schemes for approximate clustering in very large data sets. *Proc. IEEE Int. Conf. Fuzzy Systems*, pages 15–21, 2004.

[17] B. Bhanu and A. Dong. A new semi-supervised em algorithm for image retrieval. *CVPR*, 2003.

[18] M. Bilenko and R.J. Mooney. Adaptive duplicate detection using learnable string similarity measures. *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and data Mining (KDD)*, pages 39–48, 2003.

[19] A. Bouchachia and W. Pedrycz. Enhancement of fuzzy clustering by mechanisms of of partial supervision. *Fuzzy Sets and Systems*, 157:1733Ð1759, 2006.

[20] P. Salembier B.S. Manjunath and T. Sikora. *Introduction to MPEG 7: Multimedia Content Description Language*. John Wiley, 2002.

[21] J. Wang C. Aggarwal, J. Han and P. Yu. A framework for clustering evolving data streams. *Proc. Int. Conf. Very Large Databases*, pages 81–92, 2003.

[22] M. Mohri C. Cortes and A. Rostamizadeh. Two-stage learning kernel algorithms. *ICML*, pages 239–246, 2010.

[23] F. Can. Incremental clustering for dynamic information processing. *ACM Trans. Inf. Syst.*, 11(2):143–164, 1993.

[24] J. Chiang and P. Hao. A new kernel-based fuzzy clustering approach: support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems*, 11:518Ð527, 2003.

[25] M.M.C. Cortes and A. Rostamizadeh. L2 regulization for learning kernels. *UAI*, 2009.

[26] R. Caruana D. Cohn and A. McCallum. Semi-supervised clustering with user feedback. 2003.

[27] P. Hore L. Hall D. Goldgof, Y. Gu and A. Maudsley. A scalable framework for segmenting resonance images. *Signal Processing and Systems*, 54(1-3):183–203, 2009.

[28] T. Cheng D. Goldgof and L. Hall. Fast clustering with application to fuzzy rule generation. *Proc. IEEE Int. Conf. Fuzzy Systems*, pages 2289–2295, 1995.

[29] D. Klein D. Kamvar and C.D. Manning. Spectral learning. *IJCAI*, 2003.

[30] S.D. Kamvar D. Klein and C.D. Manning. From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. *Proc. of the 19th Int. Conf. on Machine Learning*, pages 307–314, 2002.

[31] I. Davidson and S. Ravi. Clustering under constraints: Feasibility issues and the k-means algorithm. *Proc. Of the 5th SIAM Int. Conf. on Data Mining (SDM)*, 2005.

[32] I. Davidson and S.S. Ravi. Agglomerative hierarchical clustering with constraints. *Proc. of the 9th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 59–70, 2005.

[33] A. Dong and B. Bhanu. Concepts learning with fuzzy clustering and relevance feedback. *Proc. Workshops on Machine Learning and Data Mining in Pattern recognition*, 2001.

[34] P. Drineas and M. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[35] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compacy well-separated clusters. *Journal of Cybernetics*, 3:32Ð57, 1973.

[36] M.I. Jordan E.P. Xing, A.Y. Ng and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems*, 15:505–512, 2003.

[37] C. Snavely F. Can, E. Fox and R. France. Incremental clustering for very large document databases: Initial marian experience. *Inf. Sci.*, 84(1-2):101–114, 1995.

[38] R. Kruse F. Hoppner, F. Klawonn and T. Runkler. *Fuzzy Cluster Analysis*. Wiley, 1999.

[39] H. Frigui and C. Hwang. Fuzzy clustering and aggregation of relational data with instance-level constraints. *IEEE Transactions on Fuzzy Systems*, 16(6):1565–1581, 2008.

[40] H. Frigui and R Krishnapuram. Clustering by competitive agglomeration. *Pattern Recognition*, 30:1223–1232, 1997.

[41] S. Mika G. Ratsch K. Tsuda K. Muller and B. Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Network*, 12:181Ð201, 2001.

[42] S. Chatterjee G. Sheikholeslami and A. Zhang. Wavecluster: A multi- resolution clustering approach for very large spatial databases. *Proc. of the 24th Int. Conf. on Very Large Databases*, pages 428–439, 1998.

[43] P.V. Gehler and S. Nowozin. Infinite kernel learning. Technical report, Max Planck Institute for Biological Cybernetics, 2008.

[44] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13:780Ð784, 2002.

[45] D. Graves and W. Pedrycz. Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study. *Fuzzy Sets and Systems*, 161:522Ð543, 2010.

[46] N. Cristianini M.I. Jordan G.R.G. Lanckriet, T.D. Bie and W.S. Noble. A statistical framework for genomic data fusion. *Bioinfomatics*, 20:2626Ð2635, 2004.

[47] P.L. Barlett L.E. Ghaoui G.R.G Lanckriet, N. Cristianini and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of machine Learning Reserach*, 5:27–72, 2004.

[48] E. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *IEEE CDC*, 1979.

[49] D. Horn H. S. A. Ben-Hur and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125Ð137, 2001.

[50] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1074–1085, 1992.

[51] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. *Proc. ACM Symp. Theory Comput.*, pages 291–300, 2004.

[52] R.J. Hathaway and J.C. Bezdek. Nerf c-means: Non-euclidean relational fuzzy clustering. *Pattern Recognition*, pages 429–437, 1994.

[53] S. Haykin. *Neural networks: Comprehensive foundation*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1999.

[54] A. Hinneburg and D. Keirn. An efficient approach to clustering in large multimedia databases with noise. *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[55] S.C.H. Hoi and W. L. S. Chang. Semi supervised distance metric learning for collaborative image retrieval. *CVPR*, 2008.

[56] F. Klawonn F. Hoppner and R. Kruse. *Fuzzy cluster analysis: Methods for classification, data analysis and image recognition*. New York: Wiley, 1999.

[57] Hsin-Chien Huang. Multiple kernel fuzzy clustering. *IEEE TFS*, 20:120–134, 2012.

[58] Y. Guan I. Dhillon and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. *ACM SIGKDD*, 2004.

[59] R Gunderson J. Bezdek, C. Coray and J. Watson. Detection and characteri- zation of cluster substructure. *SIAM*, pages 339–357, 1981.

[60] J. Li J. Z. Wang and G. Wiederhold. Simplicity: semantics-sensitive integrated matching for picture libraries. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(9):947–963, 2001.

[61] I.W. Tsang J. Zhuang and S.C.H. Hoi. A family of simple non-parametric kernel learning algorithms. *Journal of Machine Learning Research*, 12:1313–1347, 2011.

[62] A.K. Jain and M.N. Murty. Data clustering: A review. *ACM Computing Surveys*, 1999.

[63] T.C. Havens J.C. Bezdek and M. Palaniswam. Incremental kernel fuzzy c-means. *Computational Intelligence: Revised and Selected Papers from IJCCI 2010*, 2011.

[64] S. Rogers K. Wagstaff, C. Cardie and S. Schroedl. Constrained k-means clustering with background knowledge. *Proc. of the 18th Int. Conf. on Machine Learning*, pages 577–584, 2001.

[65] I. W. Tsang K. Zhang and J. T. Kwok. Maximum margin clustering made practical. *Proceedings of the 24th international conference on machine learning*, page 1119Ð1126, 2007.

[66] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: Wiley-Blackwell, 2005.

[67] J. Ke L. Hall, S. Eschrich and D. Goldgof. Fast accurate fuzzy clustering through data reduction. *IEEE Trans. Fuzzy Systems*, 11:262–269, 2003.

[68] P. Hore L. Hall and D. Goldgof. Single pass fuzzy cmeans. *Proc. IEEE Int. Conf. Fuzzy Systems*, pages 1–7, 2007.

[69] B. Larson L. Xu, J. Neufeld and D. Schuurmans. Maximum margin clustering. *Advances in Neural Information Processing Systems*, 2005.

[70] J. Sander M. Ester, H. Kriegel and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. *Proc. of the AMC SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 226–231, 1996.

[71] F. Masulli M. Filippone, F. Camastra and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41:176–190, 2008.

[72] E. Alpaydin M. Gonen. Localized multiple kernel learning. *Proceedings of the 25th International Conference on Machine Learning*, pages 352–359, 2008.

[73] S. Sonnenburg P. Laskov K-R. Muller M. Kloft, U. Brefeld and A. Zien. efficient and accurate lp-norm multiple kernel learning. *NIPS*, 2009.

[74] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press*, pages 281–297, 1967.

[75] M. Meila and J. Shi. A random walks view of spectral segmentation. *IWAIS*, 2001.

[76] M. Crucianu N. Grira and N. Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of Machine Learning Techniques for Proceedings Multimedia Content. Report of the MUSCLE European Network of Excellence*, 2004.

[77] M. Crucianu N. Grira and N. Boujemaa. Active semi-supervised clustering for image database categorization. *Proc. of the 7th ACM SIGMM Int. Workshops on Multimedia Information retrieval*, pages 9–16, 2005.

[78] M. Crucianu N. Grira and N. Boujemaa. Semi-supervised fuzzy clustering with pairwise-constrained competitive agglomeration. *IEEE Conf. on Fuzzy Systems*, pages 867–872, 2005.

[79] R. Ng and J. Han. Clarans: Amethod for clustering objects for spatial data mining. *IEEE Trans. Knowl. Data Eng.*, 14(5):1003–1016, 2002.

[80] R. Krishnapuram O. Nasraoui and A. Joshi. Relational clustering based on a new robust estimator with application to web mining. *IWWWC*, 1999.

[81] W. Pedrycz. Algorithms of fuzzy clustering with partial supervision. *Pattern Recognition Letters*, 3:13–20, 1985.

[82] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Transactions on Systems*, 27:787–795, 1997.

[83] W. Jin R. Ge, M. Ester and I. Davidson. Constraint-driven clustering. *Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 320–329, 2007.

[84] J. W. Davenport R. J. Hathaway and J. C. Bezdek. Relational duals of the c-means algorithms. *Pattern Recognition,*, 22:205–212, 1989.

[85] S.C.H. Hoi R. Jin and T. Yang. Online multiple kernel learning: Algorithms and mistake bounds. *21st Int. Conf. on Algorithmic Learning Theory*, pages 390–404, 2010.

[86] H. Frigui R. Krishnapuram, O. Nasraoui and A. Joshi. Extracting web user profiles using relational competitive fuzzy clustering. *nternational journal on artificial Intelligence Tools*, 9(4):509–526, 2000.

[87] P. H. Hart R. O. Duda and D. G. Stock. *Pattern Classification*. John Wiley and Sons, 2001.

[88] J.M. Huband R.J. Hathaway and J.C. Bezdek. A kernelized non-euclidean relational fuzzy c-means algorithm. *FUZZ-IEEE*, 2005.

[89] Y. Rubner. *Perceptual metrics for image database navigation*. Stanford University, Stanford, 1999.

[90] E. Ruspini. Numerical methods for fuzzy clustering. *Information science*, pages 319–350, 1970.

[91] A. Banerjee S. Basu and R.J. Mooney. Semi-supervised clustering by seeding. *Proc. of the 19th Int. Conf. on Machine Learning (ICML-2002)*, pages 19–26, 2002.

[92] A. Banerjee S. Basu and R.J. Mooney. Active semi-supervised for pairwise constrained clustering. *Proc. of the SIAM Int. Conf. on data Mining (SDM)*, pages 333–344, 2004.

[93] M. Bilenko S. Basu and R. J. Mooney. Comparing and unifying searchbased and similarity-based approaches to semi-supervised clustering. *Proc. of the ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, page 42Ð49, 2004.

[94] D. Klein S. D. Kamvar and C. D. Manning. From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. *Proc. Int. Conf. on Machine Learning*, pages 307–314, 2002.

[95] N. Mishra R. Motwani S. Guha, A. Meyerson and L. Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. Knowl. Data Eng.*, 15(3):515–528, 2003.

[96] R Rastogi S. Guha and K. Shim. Cure: An efficient clustering algorithm for large databases. *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pages 73–84, 1998.

[97] R. Rastogi S. Guha and K. Shim. Cure: An efficient clustering algorithm for large databases. *Inf. Syst.*, 2691):35–58, 2001.

[98] C. Cardie S. Rogers, K. Wagstaff and S. Schroedl. Constrained k-means clustering with background knowledge. *Proc. Int. Conf. on Machine Learning*, pages 577–584, 2001.

[99] C. Schafer S. Sonnenburg, G. Ratsch and B. Scholkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[100] F. Bach S.C.A. Rakotomamonjy and Y. Grandvalet. More efficiency in multiple kernel learning. *ICML*, 2007.

[101] R. Jin S.C.H. Hoi and M.R. Lyu. Learning non-parametric kernel matrices from pairwise constraints. *Proceedings of the 24th international conference on machine learning*, 2007.

[102] B. Scholkopf and A. Smola. *Learning with kernels: Support vector machines, regulization, optimization and beyond.* Cambridge, MA: MIT Press, 2002.

[103] S. Sen and R.N. Dave. Clustering of relational data containing noise and outliers. *Proc. IEEE Conference on Fuzzy Systems*, pages 1411–1416, 1998.

[104] B.U. shankar and N. Pal. An effective approach for large data sets. *Proc. Int. Conf. Fuzzy Logic, Neural Nets and Soft Computing*, 1994.

[105] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[106] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, pages 888–905, 2000.

[107] P.H Sneath and R.R. Sokal. *Numerical taxonomy*. Freeman,San Fransisco, 1973.

[108] N. Srebro and S. Ben-David. Learning bounds for support vector machines with learned kernels. *COLT*, page 169Ð183, 2006.

[109] A. Jain T. Havens, R. Chitta and R. Jin. Speedup of fuzzy and possibilistic c-means for large-scale clustering. *Proc. IEEE Int. Conf. Fuzzy Systems*, pages 463–470, 2011.

[110] R. Jin T. Havens, R. Chitta and A. Jain. Approximate kernel k-means: solution to large scale kernel clustering. *Proc. ACM SIGKDD*, 2011.

[111] B. Scholkopf T. Hofmann and A.J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36:1171–1220, 2008.

[112] R. Ramakirshnan T. Zhang and M. Livny. Birch: An efficient data clustering method for very large databases. *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pages 103–114, 1996.

[113] C. Leckie L.O. Hall T.C. Havens, J.C. Bezdek and M. Palaniswami. Fuzzy c-means algorithms for very large data. *IEEE T*, 20(6):1130–1146, 2012.

[114] UCI. Repository of machine learning databases (http://archive.ics.uci.edu/ml/datasets.html).

[115] V. Vapnik. *Statistical learning theory*. New York: Wiley, 1998.

[116] V.N. Vapnik. *The nature of statistical learning theory*. Springer, NY, USA, 1995.

[117] M. Varma and B.R. Babu. More generality in efficient multiple kernel learning. *Proceedings of the International Conference on Machine Learning*, page 134, 2009.

[118] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395Ð416, 2007.

[119] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. *Proc. of the 17th Int. Conf. on Machine Learning*, pages 1103–1110, 2000.

[120] C. Rostoker X. Wang and H.J. Hamilton. Density-based spatial clustering in the presence of obstacles and facilitators. *Proc. of the 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 446–458, 2004.

[121] C. Tomasi Y. Rubner and L.J. Guibas. A metric for distributions with applications to image databases. *Proc. 6th Conf. on Computer Vision*, pages pp. 59–66, 1998.

[122] S.J.J. Ye and J. Chen. Learning the kernel matrix in discriminant analysis via quadratically constrained quadratic programming. *SIGKDD*, 2007.

[123] Y. Ying and C. Campbell. Generalization bounds for learning the kernel. *COLT*, 2009.

[124] W.-X. Xie Z.-D. Wu and J.-P. Yu. Fuzzy c-means clustering algorithm based on kernel method. *ICCIMA*, pages 49–54, 2003.

[125] I. King Z. Xu, R. Jin and M.R. Lyu. An extended level method for efficient multiple kernel learning. *NIPS*, pages 1825–1832, 2008.

[126] O.R Zaine and C-H. Lee. Clustering spatial data in the presence of obstacles. *Proc. of the 6th Int. Symposium on Database Engineering and Applications*, pages 214–223.

[127] D.-Q. Zhang and S.-C. Chen. Fuzzy clustering using kernel method. *ICCA*, 2002.

[128] D.-Q. Zhang and S.-C. Chen. Clustering incomplete data using kernel-based fuzzy c-means algorithm. *Neural Processing Letters*, 18:155Ð162, 2003.

[129] Y. Zho and G. Karypis. Hierarchical clusterin algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10:141–168, 2005.

CURRICULUM VITAE

Name: Naouel Baili

Address: CECS Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

Education:

Ph.D., Computer Science & Engineering
May 2013
University of Louisville, *Louisville, Kentucky*

M.S., Computer Engineering
July 2007
Institute of Applied Sciences and Technologies, *Sousse, Tunisia*

B.Eng., Polytechnic Engineering
June 2005
Tunisia Polytechnic School, *Tunis, Tunisia*

Journal Publications:

1. **N. Baili**, and H. Frigui, "*Unsupervised and semi-supervised Relational Fuzzy Clustering with Multiple Kernels*", (under preparation).

2. **N. Baili**, and H. Frigui, "*Relational Fuzzy Clustering with Multiple Kernels for Very Large Data*", (under preparation).

Conference Publications:

1. **N. Baili**, and H. Frigui, "*Incremental Fuzzy Clustering with Multiple Kernels*", European Signal Processing Conference 2013 (under review).

2. **N. Baili**, and H. Frigui, "*Semi-Supervised Clustering with Cluster-Dependent Multiple Kernels*", International Conference on Information, Intelligence, Systems and Applications 2013 (under review).

3. **N. Baili**, and H. Frigui, "*Fuzzy Clustering with Multiple Kernels in Feature Space*", IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2012), Australia, June 2012.

4. **N. Baili**, and H. Frigui, "*Fuzzy Clustering with Multiple Kernels*", IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011), Taiwan, June 2011.

5. **N. Baili**, and H. Frigui, "*Relational Fuzzy Clustering with Multiple Kernels*", ICDM Workshop on Mining Multiple Information Sources, Canada, December 2011.

6. A.A. Mohamed, D. DSouza, **N. Baili**, and R.V. Yampolskiy "*Avatar Face Recognition Using Wavelet Transform and Hierarchical Multi-scale LBP*", International Conference on Machine Learning and Applications (ICMLA 2011), Hawaii, December 2011.